Model Predictive MRAS Estimator for Sensorless Induction Motor Drives

Yaman Zbede
B.Sc., M.Sc.

A thesis submitted for the degree of
Doctor of Philosophy

October, 2016

School of Electrical and Electronic Engineering

Newcastle University
United Kingdom

ABSTRACT

The project presents a novel model predictive reference adaptive system (MRAS) speed observer for sensorless induction motor drives applications. The proposed observer is based on the finite control set-model predictive control principle. The rotor position is calculated using a search-based optimization algorithm which ensures a minimum speed tuning error signal at each sampling period. This eliminates the need for a proportional integral (PI) controller which is conventionally employed in the adaption mechanism of MRAS observers. Extensive simulation and experimental tests have been carried out to evaluate the performance of the proposed observer. Both the simulation and the experimental results show improved performance of the MRAS scheme in both open and closed-loop sensorless modes of operation at low speeds and with different loading conditions including regeneration. The proposed scheme also improves the system robustness against motor parameter variations and increases the maximum bandwidth of the speed loop controller.

However, some of the experimental results show oscillations in the estimated rotor speed, especially at light loading conditions. Furthermore, due to the use of the voltage equation in the reference model, the scheme remains sensitive, to a certain extent, to the variations in the machine parameters. Therefore, to reduce rotor speed oscillations at light loading conditions, an adaptive filter is employed in the speed extraction mechanism, where an adaptation mechanism is proposed to adapt the filter time constant depending on the dynamic state of the system. Furthermore, a voltage compensating method is employed in the reference model of the MP-MRAS observer to address the problems associated with sensitivity to motor parameter variation. The performance of the proposed scheme is evaluated both experimentally and by simulation. Results confirm the effectiveness of the proposed scheme for sensorless speed control of IM drives.

ACKNOWLEDGMENT

I would like to express my gratitude to my supervisors dr. Dave Atkinson and dr. Shady Gadoue for the useful comments, remarks and engagement through the process of this work.

My parents, *Bakri* and *Khadija*, and my brothers, *Hosam* and *Mohanad*, receive my deepest gratitude and love for their dedication and the many years of support during my whole life that provided the foundation for this work

I would like to thank my wife *Nouran* for her understanding and love during the past few years. Her support and encouragement was in the end what made this work possible..

There is my son *Ryan* who has grown up watching me study and has given me much happiness.

I would like also to acknowledge the assistance provided by all of my friends and colleges in the school of electrical and electronic engineering in Newcastle University.

Finally I would like to thank my teachers and friends back in Syria for their support despite all the difficulties.

Yaman Zbede, October 2016

TABLE OF CONTENT

TABLE OF FIGURES

TABLE OF TABLES

CHAPTER 1

Introduction and Scope of the Thesis

1.1 Introduction

Formerly, controlled direct current (DC) machine drives were used for high performance applications due to their simple control arrangement. In contrast, alternating currents (AC) machines are low-cost, compact, and require less maintenance in comparison with the DC machines, but they require more complex control [1, 2]. However, due to recent developments in the power electronics and signal processing, induction motors (IM) and permanent magnet synchronous machine (PMSM) drives are replacing the DC machines drives in many applications especially those require high dynamic performance. Nowadays, AC drives have been widely applied in many industrial applications. This is especially the case for IM drives, due to the fact that they are the most simple, reliable, efficient and low cost compared to other machines [2]. Many drive techniques have been used with AC machines that range from low performance to high performance control.

Constant-volts-per-hertz is a well-known approach that is used to control induction machine drives in low performance applications. This technique controls the rotor speed by changing the frequency of the stator voltage. The amplitude of the applied voltage is adjusted in proportion to the stator frequency in order to maintain constant flux linkage in the machine and hence constant electromagnetic torque [3]. This method offers an acceptable dynamic performance in applications where accurate speed control is not required such as for pumps and fans [4].

High performance IM drives usually uses vector control (VC) or direct torque control (DTC) [2]. These techniques allow the control of the instantaneous machine magnetic flux and torque during both the transient and steady state operating conditions. Consequently, these techniques allows the control of the IM so that a similar dynamic performance of the DC machine drive can be achieved.

VC has established an increased acceptance in industry especially after the rapid development of power electronic converters and digital signal processors (DSP). Fig. 1.1 shows a block diagram of an IM drive system.

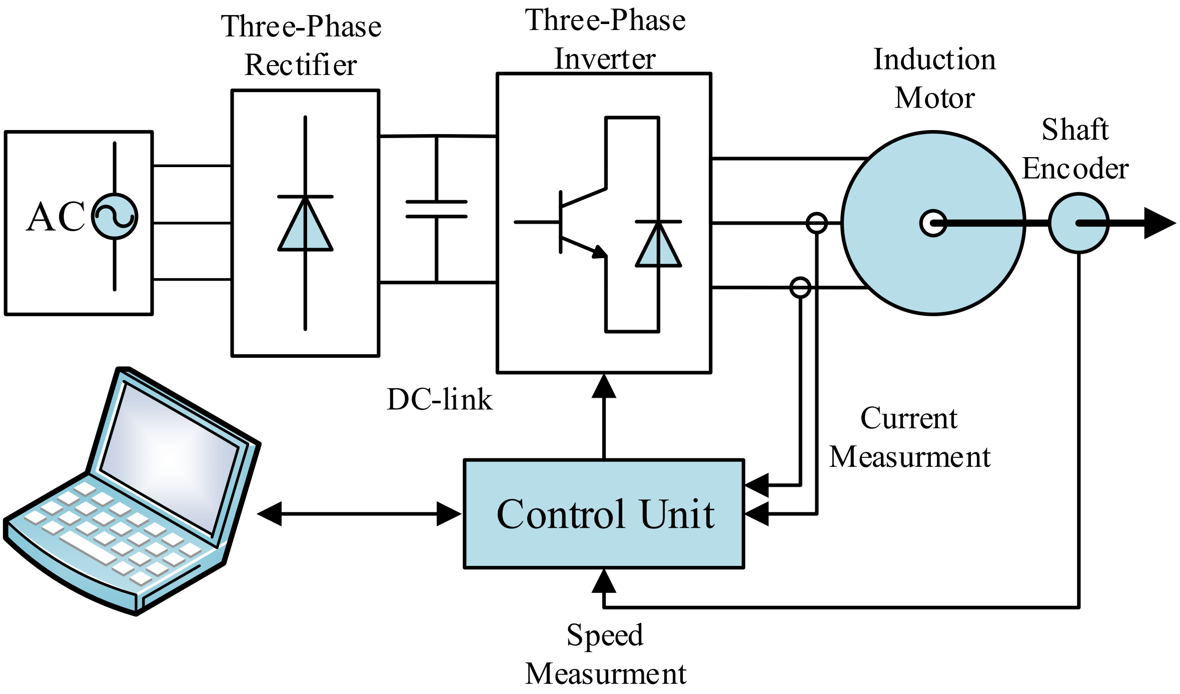


Fig. 1.1 Block diagram of an IM drive system

Over the last three decades, a new technology which is referred to as "sensorless control" has found acceptance in many IM drive applications. This technology eliminates the need for the speed sensor by using the current and voltage measurements at the terminal of the machine instead to estimate the rotor speed [2].

1.2 Speed Sensorless Control of IM Drives

Sensorless control has found wide acceptance by the engineering community since it reduces the system cost and maintenance requirements and increases the drive reliability [1, 2]. In addition, applying a speed sensor can contain some risks when it comes to operate in a hostile environment.

Sensorless control of electric drives has been successfully applied in high and medium speed regions. Conversely, at low speeds when the stator frequency become close to zero most sensorless systems fail which make them improper for applications such as traction and cranes where the drive is required to maintain the full torque down to zero speed [5]. Many attempts have been made so far to extend the operational region of sensorless systems in the low speed region [6, 7].

During the last thirty years, many different sensorless control schemes have been investigated [1, 2]. These schemes can be categorised into two distinctive classes: spectral analysis and model based schemes. Fig 1.2 shows a flowchart of the classification of the different sensorless control schemes.

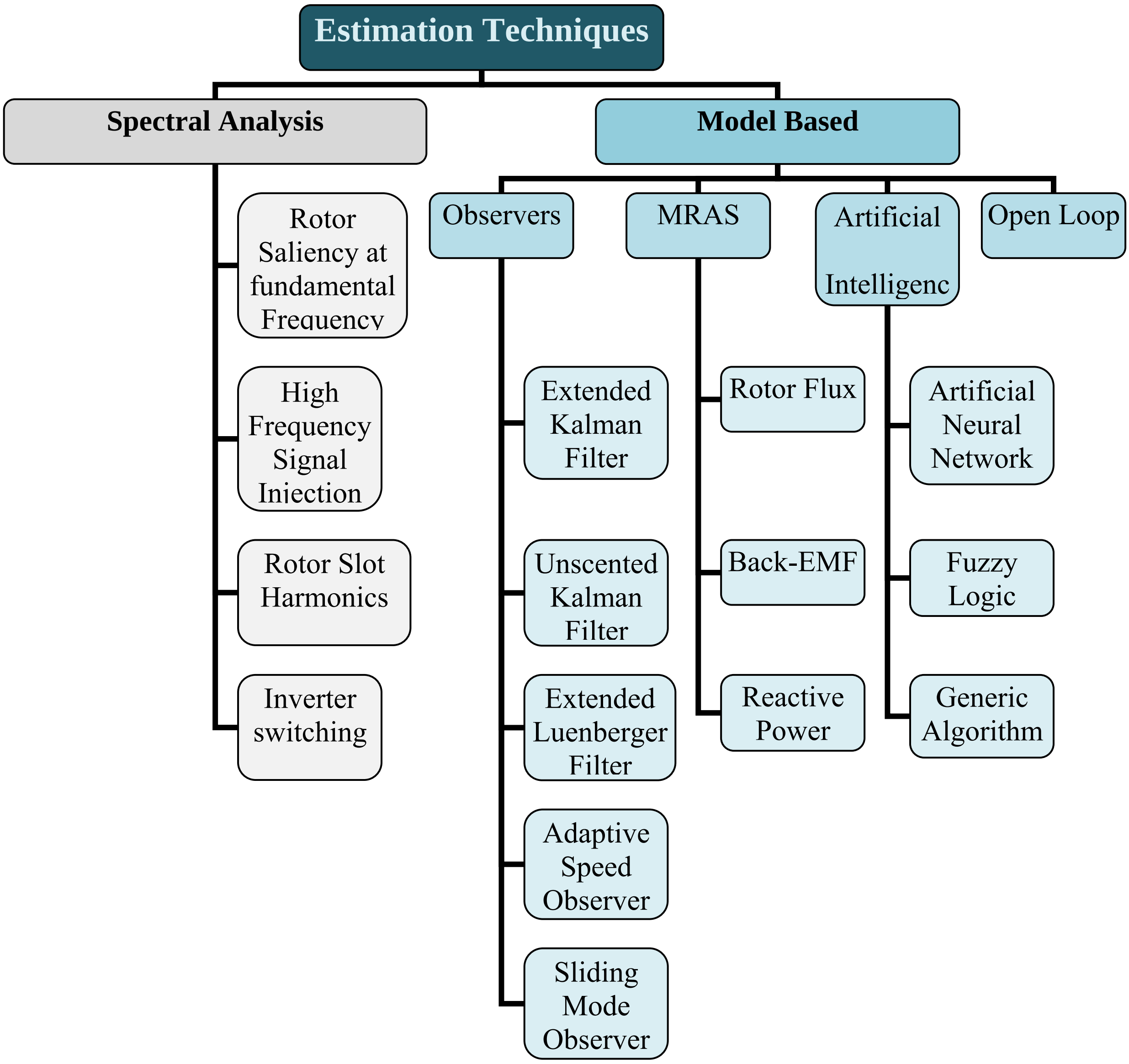


Fig. 1.2 Classification of sensorless control techniques

1.2.1 Spectral analysis sensorless schemes

In spectral analysis strategies, the machine windings are injected by high frequency low amplitude signals. These signals aret modified by the rotor orientation and are then processed to yield the rotor position. This signal injection allows extraction of the rotor position information without affecting the machine fundamental performance [1].

Using spectral analysis methods, provides an accurate speed and position estimation even at very low and zero speed operation as it does not depend on the machine parameters and the induced voltage in the rotor circuit [3]. However, these methods tend to be very sophisticated and the design is usually specific to a particular machine which makes it difficult to produce a general design [8]. Furthermore, these methods may introduce torque ripple, audible noise and vibration on the machine shaft [9].

1.2.2 . Model-based sensorless strategies

These strategies use the machine mathematical model along with the stator instantaneous currents and voltages to estimate the motor speed, assuming a sinusoidal distribution of the flux around the air gap and negligible space harmonics [1].The general structure of the model-based sensorless control drive system is illustrated in Fig. 1.3 [1].

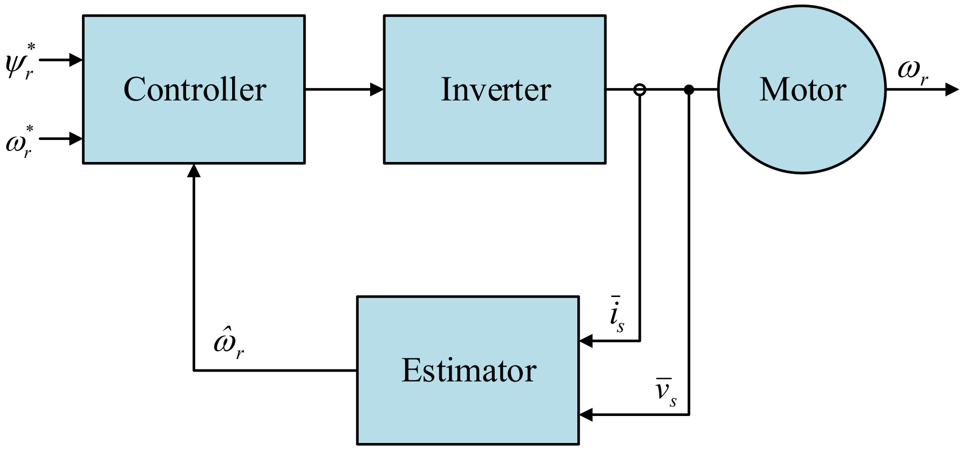


Fig. 0.3 Block diagram of general model-based sensorless control drive system [1]

Model-based strategies are capable of delivering accurate motor speed and rotor flux estimates down to 2% of the rated speed [1]. However, they fail at lower speeds due to many different factors. This includes the increased sensitivity to the machine parameters variation, reduced signal to noise ratio in the acquired voltages and currents signals and errors due to inverter nonlinearity [1, 6, 8]. In addition, at zero frequency operation the back-EMF signal induced in the rotor circuit disappears and therefore the rotor position information is lost [1, 6, 8].

Model-based sensorless strategies are basically classified into open loop and closed loop estimators. The close loop estimators (observers) employ a feedback term which is calculated as a difference between estimated and measured quantities to correct the observer estimates. Model reference adaptive system (MRAS) observers are closed loop estimators which are commonly used for sensorless control applications, due to their simple structure and ease of application. These observers are explained in detail in the following chapters.

1.3 Predictive Sensorless Control for IM Drives

Predictive control (PC) was developed in the early seventies, and is now used in power electronics and electric drive applications as an alternative to the conventional current and speed controllers [10]. Although PC techniques for electric drive have shown better performance in comparison with the classical direct torque control (DTC) and field oriented control (FOC) [11, 12], their application was limited because of their high computational requirements [13, 14]. However, recent developments in microcontrollers and digital signal processors (DSP) has led to an increased interest in PC [15, 16] as the computational barrier of their application has been removed.

Recently, PC has been employed for sensorless control of electric drives in applications including permanent magnet machines [17-20], and induction machines [21-27]. Applying predictive control has shown an improved performance of the sensorless drive system by reducing the noise effects in the measured signals and improving the system robustness against motor parameter variation.

1.4 Scope and Novelty of the Thesis

In this thesis, a novel speed estimator for sensorless vector control IM drives is introduced to address the problems associated with the MRAS adaption mechanism design. The finite control set model predictive control (FCS-MPC) concept is incorporated in the MRAS estimator design. In this scheme, the adaptation mechanism is based on solving an optimization problem with the objective of minimizing the speed tuning error signal of the

MRAS estimator over a finite number of rotor position angles. A rotor position search algorithm is developed to ensure that the optimal position is obtained at each sampling time. The computational complexity of the proposed scheme is evaluated and a modified method is employed to reduce its execution time to make it suitable for practical implementation. The performance of the proposed predictive estimator is experimentally tested using a 2.2-kW IM drive which employs FOC as the motor control strategy. A detailed comparison between the proposed scheme and the classical rotor flux MRAS estimator has been carried out. Results show the superior performance of the proposed scheme at different low-speed operating conditions including regeneration and improved robustness against motor parameter variations.

A novel adaptive filter-based speed extraction method has also been introduced to reduce the oscillations in the rotor speed at light loading conditions. An adaptation mechanism is proposed to adapt the filter time constant depending on the dynamic state of the system. Furthermore, a voltage compensating method is employed in the reference model of the model predictive MRAS observer to solve problems associated with sensitivity to motor parameters variation. The performance of the proposed schemes is experimentally evaluated. Results confirm the effectiveness of the proposed scheme for sensorless speed control of IM drives.

1.4.1 Research objectives

The core objectives of this research can be briefly summarized:

1. Develop a novel speed estimator for sensorless control of the electric machine. This replaces the PI controller in the adaptation mechanism and the adaptive model in the classical MRAS observer by a predictive model.

2. Reduce the computational complexity of the proposed predictive scheme to reduce its execution time and to make it suitable for practical implementation.

3. Reduce the oscillations in the estimated rotor speed signal at light loading conditions by introducing a novel adaptive filter-based speed extraction method.

4. Solve the problem of sensitivity to motor parameter variation by introducing a voltage compensating method to the reference model.

5. Examine the effectiveness of the introduced scheme by simulation and experimental evaluation under different speeds and loading conditions.

1.4.2 Contribution to knowledge

1. A novel speed observer is introduced which is based on the finite control set-model predictive control principle.

2. The rotor position is calculated using a search-based optimization algorithm which ensures a minimum speed tuning error signal at each sampling period.

3. The new speed observer eliminates the need for a proportional-integral (PI) controller in the adaptation mechanism.

4. The proposed scheme improves the system robustness against motor parameter variations and increases the maximum bandwidth of the speed loop controller.

5. The proposed scheme reduces the oscillations in the estimated rotor speed at all different loading conditions.

1.5 Publications

Publications resulting from this work are listed here:

- Y. B. Zbede, S. M. Gadoue, and D. J. Atkinson, "Model Predictive MRAS Estimator for Sensorless Induction Motor Drives," IEEE Transactions on Industrial Electronics, vol. 63, pp. 3511-3521, 2016.

- Y. Zbede, S. M. Gadoue, D. J. Atkinson, and M. A. Elgendy, "Predictive sensorless control of induction motor drives," in Industrial Technology (ICIT), 2015 IEEE International Conference on, 2015, pp. 2339-2344.

1.6 Thesis Layout

In Chapter 2, a literature review of different sensorless estimation schemes is provided with an emphasis given to MRAS observers. Chapter 3 explains the modelling of IMs, the principle of vector control and rotor flux-based MRAS mathematical model. In Chapter 4, a brief introduction to predictive control is given followed by a detailed description of the proposed predictive observer. Chapter 5 covers the simulation result. Chapter 6 gives a description of the experimental rig used in this project. Chapter 7 covers the experimental result. Chapter 8 is the conclusion and the future work.

CHAPTER 2

Model Reference Adaptive Systems Speed Observers – A Literature Review

2.1 Introduction

Using optical or mechanical speed sensors to measure the speed in an electric drive, increases the system size, weight and complexity and reduces the reliability by increasing the system maintenance requirements. Therefore, the machine terminal voltages and currents have been used to estimate the speed to overcome these weaknesses. These systems are referred to as sensorless drive systems, and are gaining increased popularity especially in the applications that are required to operate in hostile environments.

Over the last three decades, a lot of efforts have been applied to the development of sensorless control of electric drives and different strategies have been investigated. Among these strategies, model reference adaptive systems (MRAS) can be considered the most common, because of its simple structure and relatively low computational requirements. However, MRAS-based sensorless drives still have many problems especially when it comes to operating at and around zero speeds and full loads. Consequently, a great deal of work has been done to improve the performance in this operating region.

In this chapter, a review of the different model-based speed estimators applied to sensorless control of electric drives is provided. A focus is given to the MRAS-based observers, the problems associated with these estimation strategies and the suggested solutions.

2.2 Model-Based Speed Observers

Many speed estimators have been introduced that depend on the machine mathematical model to calculate the rotor speed. These estimators are basically classified into two different types; open loop and closed loop estimators [8]. The difference between these two types is the existence of the feedback term in the closed loop estimator which is used to correct the speed estimation [2].

In open loop estimators, the accuracy of the machine mathematical model plays a key role in estimating the speed [28]. Therefore, these estimators suffer many problems like the sensitivity to the machine parameter variations and the noise in the measured signals especially at low speeds[28]. On the other hand, the closed loop estimators (which are also called speed observers) use a correcting term. This takes advantage of the error between measured and estimated quantities to rectify the error in the speed estimation which leads to a better-quality performance and an improved robustness [8].

In sensorless control of electric drives, several closed loop speed observers have been applied. These include MRAS observers [29], extended Kalman filter (EKF) observers [30, 31], sliding mode (SM) observers [32, 33] and artificial intelligence (IA) based observers [34].

EKF observers for sensorless control of IMs have been widely applied [35-37]. These observers employ a mathematical model of the IM which considers the noise in the plant and the measured signal and also takes into account the inaccuracy in the model. The EKF observer has been successfully applied for operation at low speed [2], but it suffers many problems like the high computational demand, the instability as a result of linearization and the absence of tuning and design criteria [38]. Furthermore, a problem of biasing can take place as a result of an improper adjustment of the covariance matrices or because of a mismatch between the assumed characteristics of the stochastic noise model and the real noise [1, 2, 38]. Recently, many researchers have focused on optimizing the EKF performance to solve the problem of the covariance matrix tuning to obtain an improved performance. Shi, Chan, et al. [37] proposed a genetic algorithm method to optimize tuning the plant and noise covariance matrices, and Buyamin, et al. [39] proposes a simulated algorithm to optimize the covariance matrix tuning process.

SM observers have been applied to estimate the rotor position in [9, 40, 41], and the PI adaption mechanism has been replaced by a sliding mode controller (SMC) in [42, 43]. Although this scheme is shown to improve the estimator dynamic response, it causes a considerable amount of chattering in the estimated speed signal, and a low-pass filter is needed to smooth out the estimated rotor speed. In [44], another solution was proposed where the PI controller is replaced by a fuzzy logic (FL)-based adaption mechanism. This scheme shows improvement in the estimator dynamic response, but the computational complexity of the FL controller is the main drawback.

Among all other speed observers, MRAS-based observers have gained great popularity because of their relative simplicity and ease of application [7, 45-47]. A comparison between MRAS and EKF observers was provided in [48]. The results in [48] shows that both observers were affected to a similar degree during the steady state by the stator resistance variation. They also show that the EKF had a poorer transient performance at low speed, and a higher computational demand which is up to 20 times more in comparison with the MRAS observer.

2.3 Model Reference Adaptive Systems

The adaptive control system is defined as "*a control system that can modify its behaviour in response to changes in the dynamics of the process and the character of the disturbances*" [49]. Among the different adaptive control strategies, the model reference adaptive control (MRAC) has been widely applied for control and parameter estimation in electric drive applications [49]. Fig. 2.1 shows a block diagram of an MRAC system.

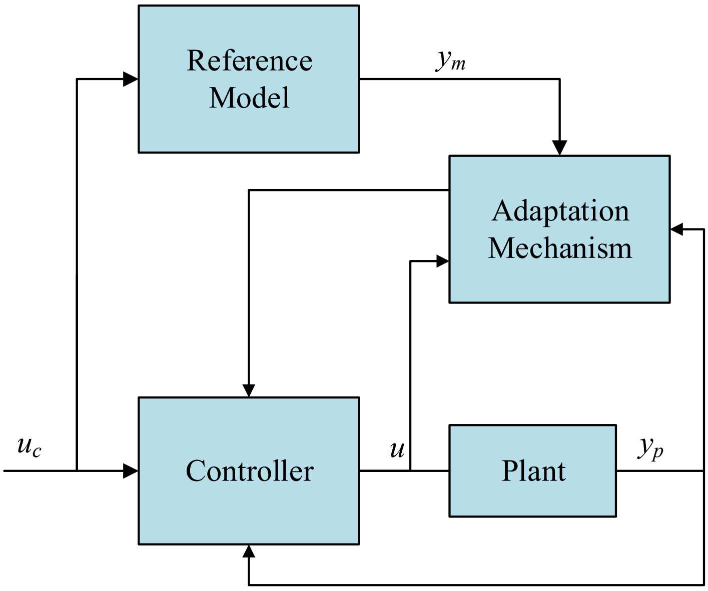


Fig. 2.1 Block diagram of MRAC system

In the block diagram, the reference model is designed to provide the desired performance of the plant y_m for a given command signal u_c. The reference model output y_m is compared with the plant real output y_p and the error is fed to an adaptation mechanism. This adaptation mechanism works by adjusting the controller parameters until the error between the plant and the reference model outputs is driven to zero.

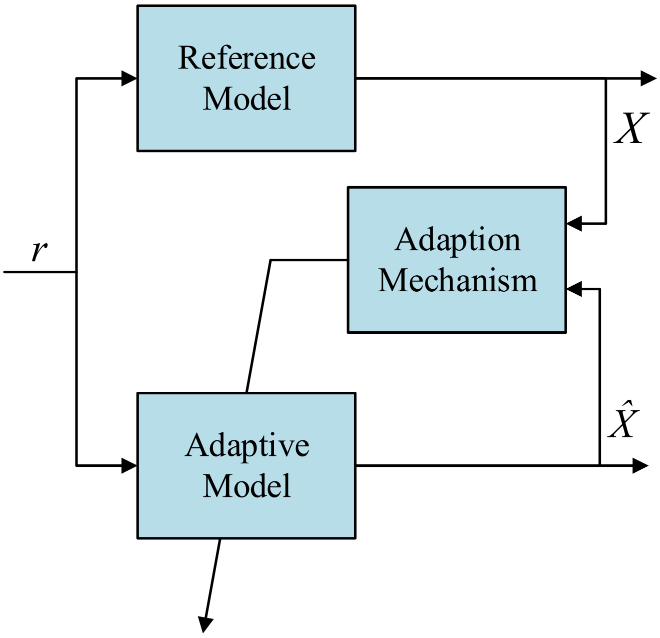


Fig. 2.2 Block diagram of MRAS observer

A similar mechanism has been applied to estimate system parameters and states in the so called MRAS estimators. MRAS estimators have been widely used for sensorless control of electric drives due to their simplicity and ease of application [50]. The first MRAS observer was proposed in [51]. This observer employs two separate sub models which are called the reference model and the adaptive model. Similarly to MRAC, the reference model is designed to provide the desired value of the state vector X, while the adaptive model uses a completely different way to find an estimate of the same state vector $\hat{X}$. The reference and the adaptive models outputs are compared and the error is fed to an adaptation mechanism, which adjusts the adaptive model parameters until the error is driven to zero. Fig. 2.2 shows the block diagram of the basic MRAS structure [51].

Depending on the states that form the error vectors, different types of MRAS observers have been introduced. The most common ones are: rotor flux-based MRAS observers [2, 46, 52, 53], back EMF-based MRAS observers [7, 54], reactive power-based MRAS observers [54-57] and more recently, electromagnetic torque- based MRAS observers [58].

2.3.1 Rotor flux-based MRAS observers

The first rotor flux based observer was introduced by *Schauder* [59] in 1989. The reference and the adaptive sub-systems independently calculate the rotor flux components in a stationary reference frame. The difference between the two sub-system estimates is then used to find the rotor speed. This observer will be studied in detail in the next chapter.

Rotor flux MRAS observers have been well investigated over the last two decades, and the superior performance of these observers has been proven over a wide range of speeds [2, 46]. However, this scheme suffers many problems at low speeds [44, 46], This includes the pure integration problem [2], the sensors data acquisition problem, and the inverter nonlinearity [6, 44] .

2.3.2 Back EMF-based MRAS observers

To avoid the problem of the pure integration in the rotor flux-based MRAS observers, *Peng* and *Fukao* suggested a new MRAS strategy in [54] which is based on calculating the back EMF signals.

In the reference model, the stator voltage equations in the stationary reference frame are used to obtain the back EMF components. This leads to:

$$e_{sD} = \frac{L_m}{L_r} p\psi_{rd} = v_{sD} - R_s i_{sD} - \sigma L_s p i_{sD} \tag{2.1}$$

$$e_{sQ} = \frac{L_m}{L_r} p\psi_{rq} = v_{sQ} - R_s i_{sQ} + \sigma L_s p i_{sQ} \tag{2.1}$$

Where e_{sD} , e_{sQ} are the back EMF components, v_{sD}, v_{sQ} are the stator voltage components, i_{sD}, i_{sQ} are the stator current components, ψ_{rd}, ψ_{rq} are the rotor flux linkage components, L_m is the mutual inductance, L_r is the rotor self-inductance, L_s is the stator self-inductance, R_s is the stator resistance and σ is the leakage coefficient.

From (2.1) and (2.2) it can be noticed that the back EMF components are calculated without any integration which solves the pure integration and DC drift problems in the rotor flux-based MRAS.

The adaptive model equations are obtained from the rotor circuit voltage equations in the stationary reference frame, and these equations are:

$$\hat{e}_{sD} = \frac{L_m}{L_r} p\hat{\psi}_{rd} = \frac{L_m}{L_r}\left(\frac{L_m}{T_r} i_{sD} - \frac{1}{T_r}\hat{\psi}_{rd} - \hat{\omega}_r\hat{\psi}_{rq}\right) \tag{2.3}$$

$$\hat{e}_{sQ} = \frac{L_m}{L_r} p\hat{\psi}_{rq} = \frac{L_m}{L_r}\left(\frac{L_m}{T_r} i_{sQ} - \frac{1}{T_r}\hat{\psi}_{rq} + \hat{\omega}_r\hat{\psi}_{rd}\right) \tag{2.4}$$

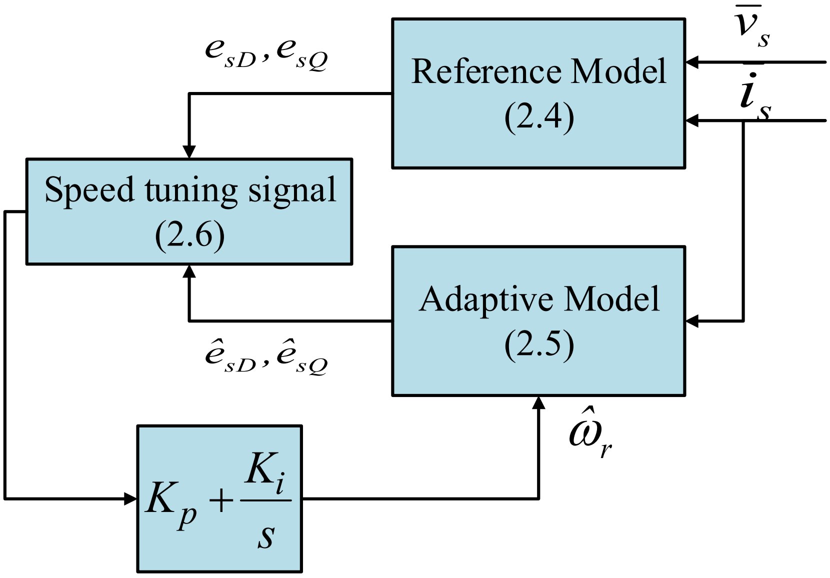


Fig. 2.3 Block diagram of the back EMF-based MRAS observer

The speed tuning signal in the case of the back EMF-based MRAS observer becomes the angular difference between the reference and the adaptive back EMF components as in the following equation:

$$\varepsilon_\omega = (e_{sQ}\hat{e}_{sD} - e_{sD}\hat{e}_{sQ}) \tag{2.5}$$

As has been discussed, using the back EMF MRAS instead of the rotor flux solves the problem of the pure integration in the reference model. However, it has been reported in [7] that the scheme has a stability issue at low speeds. It also has been claimed in [60] that this scheme has a poor dynamic performance at low speeds and low immunity toward the measurement noise duo to the stator current differentiation in the reference model.

2.3.3 Reactive power-based MRAS observers

This scheme was introduced by *Peng* and *Fukao* in [54], and it depends on the reactive power equations in estimating the rotor speed.

The reactive power can be calculated by using the following equation:

$$q = \bar{e} \times \bar{i}_s = \bar{i}_s \times (v_s - R_s \bar{i}_s - \sigma L_s p \bar{i}_s) \tag{2.6}$$

Because $\bar{i}_s \times \bar{i}_s = 0$, (2.6) becomes:

$$q = \bar{i}_s \times v_s - \sigma \cdot L_s \cdot \bar{i}_s \cdot p \bar{i}_s \tag{2.7}$$

or:

$$q = i_{sD} v_{sQ} - i_{sQ} v_{sD} - \sigma L_s (i_{sD} p i_{sQ} - i_{sQ} p i_{sD}) \tag{2.8}$$

The adaptive model equations are found similarly by using:

$$\hat{q} = \hat{\bar{e}} \times \bar{i}_s = \bar{i}_s \times \frac{L_m}{L_r} \left(\frac{L_m}{T_r} \bar{i}_s - \frac{1}{T_r} \hat{\bar{\psi}}_r + \hat{\omega}_r \hat{\bar{\psi}}_r \right) \tag{2.9}$$

or:

$$\hat{q} = \frac{L_m}{L_r} \left(\frac{1}{T_r} (\hat{\psi}_{rq} i_{sD} - \hat{\psi}_{rd} i_{sQ}) + \hat{\omega}_r (\hat{\psi}_{rd} i_{sQ} - \hat{\psi}_{rq} i_{sD}) \right) \tag{2.10}$$

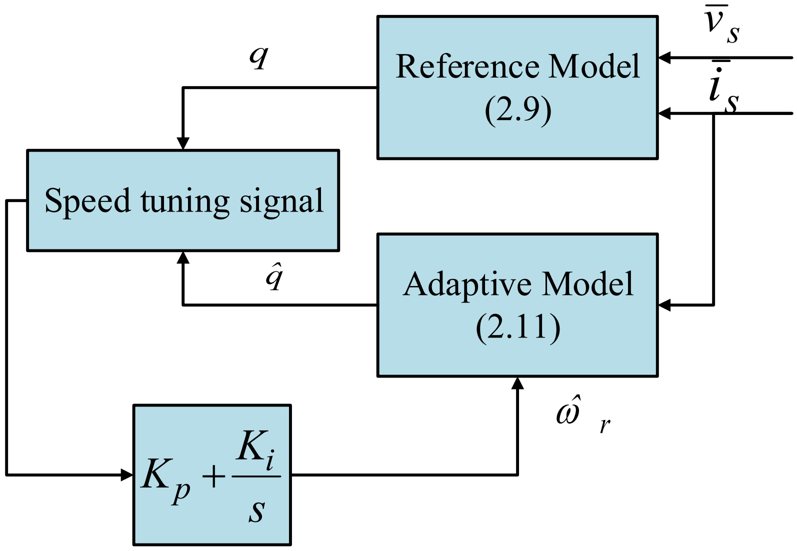


Fig. 2.4 Block diagram of the reactive power-based MRAS observer

From (2.9) it can be noticed that by using the reactive power equation, the integration and the stator resistance dependant term in the reference model disappears to solve the problems of the pure integration and the sensitivity to R_s variation. However it was mentioned in [7, 61, 62] that the scheme suffers a problem of instability at certain operation conditions in the regeneration region. Also the reactive power almost vanishes at and around the zero speed operation as the back EMF approaches zero [63].

2.4 The Problems and the Proposed Solution for MRAS-based Observers of IM Drives

It has been mentioned previously that the MRAS-based observers suffers many problems that limit their usage at and around zero speeds. In this section these problems are briefly summarised with solutions suggested in the literature.

2.4.1 The pure integration problem

This problem is particularly related to the rotor flux- based MRAS observers, and it occurs because of the need for pure integration in the voltage model [2, 6, 64]. To solve this problem, many solutions have been proposed. In [5, 65, 66], the integrator was replaced by a low pass filter (LPF) with a very low cut-off frequency. This solution is very effective for high and medium speeds operation, but at low speeds because the filter phase shift and attenuation becomes more significant, the system performance starts to deteriorate [2, 43, 67]. In [67] the integrator was replaced by a programmable cascaded small time constant LPF that behaves like an integrator in terms of gain and phase shift. However, it was found that the high speed performance deteriorates as the number of filters is increased. In [68] the pure integrator was replaced by a zero-drift feedback integrator which was able to provide a satisfactory performance down to 0.2 Hz, but then again, the system becomes unstable at lower operating frequencies. In [64, 69] the voltage model was completely replaced by a different flux model to remove the need for the pure integration, but this was at the cost of the significantly increased complexity. A closed loop flux observer was introduced in [59, 63] in which a coupling controller is employed between the reference and adaptive model outputs. The method shows a satisfactory performance at low speeds but as the speed approaches zero the scheme starts to lose its stability. Although this problem was solved by deploying the machine mechanical model, this approach requires an accurate determination of the system mechanical parameters which increases the system is complexity.

2.4.2 The problem of inverter nonlinearity and voltage acquisition

In model based sensorless techniques, the knowledge of the voltage signals across the machine terminals is essential. In the literature, the voltage has been obtained either by applying voltage sensors at the machine terminals or by using the modulation index waveforms available within the microcontroller.

Using the measured voltage directly as an input to the estimator model is not straight forward since the voltage signals must be sampled at a very high sampling frequency to avoid aliasing [6]. Alternatively, a low pass filter must be used to eliminate the high order harmonics. The filter-based scheme only works well at high and medium operating speeds where the effect of the filter phase error and gain is negligible. To solve this problem, a synchronous integrator-nonlinear filter is used to replace the classical filter, where the measured voltage is integrated over one PWM sample and then the integrator is reset at the end of the sample. This allows measurement of the actual voltage-seconds over one sampling period [70]. Although it has been shown that this method can produce satisfactory performance at low speeds, there is still a reluctance to use voltage sensors in industrial applications [9].

Replacing the measured voltages by the modulation signals available within the controller is popular since these signals are free of any PWM harmonics. However, at low speeds these signals deviate considerably from the actual voltages because of the inverter nonlinearity and the effect of the dead time imposed by the controller. In [65] a nonlinear model of the inverter was proposed which considers the power devices internal impedance and threshold voltage. A compensation for the dead-time effect was proposed in [71] which depends on the reference voltage and current signals to compensate for the effect on the stator voltage measurements during the dead time intervals.

2.4.3 The problem of motor parameter variation

As with any other model based observer schemes, the MRAS observers are very much affected by the machine parameter variations that can occur during operation. This includes variation in the rotor and the stator resistances due to changes in operating temperature and frequency. Also the mutual inductance can vary significantly with the magnetic saturation.

The main cause of the temperature variation in the machine is due to copper loss, magnetic loss and mechanical loss. These losses can vary with the loading condition, the flux level and the operating speed. Applying a temperature sensor to measure the machine operating temperature or using a thermal model to predict the temperature variation is not precise since the relationship between the resistance variation and the temperature is complex in most cases [72].

As it has already been mentioned, the stator resistance changes due to temperature variation, and this change can reaches up to 50% of the original value [1, 8]. That can cause a very serious problem especially at low speeds because the value of the applied voltage becomes very small and comparable to the voltage drop across the stator resistance [3, 60]. Consequently, it is required to adapt the stator resistance continuously [73] to increase the estimation accuracy during steady state operation and to maintain the system stability at low speeds [60, 72].

Different solutions have been proposed that provide a combined stator resistance and rotor speed estimation [7, 62, 73-75]. In [74], Zhen and Xu proposed a mutual MRAS system that can simultaneously identify both the rotor speed and the stator resistance. The reference and the adaptive models are interchangeable, which means the two models switch their roles to identify the stator resistance. This leads to a problem which is that during the rotor identification phase there is no speed information fed to the drive [75].

To address the problem in [74], Vasic *et al.* [75] proposed a new estimation method which deploys two parallel MRAS observers that provide simultaneous estimations of the rotor speed and the stator resistance. Unlike [74], this method does not require the reference and the adaptive models to interchange their roles and it estimates during the transient operation. This scheme succeeded to obtaining accurate estimations at 0.5, 1 and 4Hz operating frequencies with ±20% variation in the stator resistance.

Another method was proposed in [62] for combined stator resistance and rotor speed estimation. This method, exploits the fact that the rotor speed is dependent on the q-axis component of the rotor flux represented in a synchronous reference frame. Also the stator resistance is dependent on the d-axis rotor flux component. The scheme was evaluated at different speeds and values of stator and rotor time constants. The results show that the rotor time constant variation caused a steady state error in the estimated rotor speed but without affecting the system stability.

A combined MRAS stator resistance and speed estimator was employed in [7] which is based on back-EMF estimation. A Routh–Hurwitz-based criterion was applied to design the estimator gains to deliver stable stator resistance and rotor speed estimation even at low operating speeds. A frequency injection method was used at zero operation frequency to estimate the rotor speed. The results show stable operation at different speeds and loading conditions. However the scheme showed poor transient performance and oscillations in the speed during the steady state.

Kubota *e al.* [76] proposed a method to identify the motor speed and the rotor resistance simultaneously using a full-order adaptive flux estimator. A new technique for rotor resistance estimation was proposed which depends on injecting an AC signal into the field current demand. The scheme, however, does not discuss the impact of variation of the other motor parameters which are part of the flux estimator including the rotor and the stator self-inductances.

2.4.4 The problem with the MRAS adaptation mechanism

Generally, a fixed-gain proportional–integral (PI) controller is employed in the adaptation mechanism of MRAS schemes to produce the estimated position or speed. This is because of its simple structure and ability to generate a satisfactory performance over a wide range of speeds. However, at low speeds, inverter nonlinearities and machine parameter variation become more dominant. As a result, the fixed-gain PI may not be able to maintain the system stability or at least to provide the required performance. Moreover, tuning of the PI gains is not an easy task and little effort has been devoted to address this problem in the literature.

Various solutions to offer alternative approaches to the design of the adaptation mechanism for MRAS estimators have been discussed in the literature. These solutions have focused on replacing the conventional fixed-gain PI adaption mechanism with more advanced algorithms.

Shiref *et al.* [77] investigated experimentally the influence of the PI controller gains on the estimated speed. The results revealed that by applying high PI gains, the estimated speed becomes noisy and it starts to contain some harmonics which are synchronised with the stator's electrical frequency. An adaptive-PI based adaptation mechanism was suggested to avoid the noise in the estimated speed. The performance of proposed and classical adaptation mechanisms was experimentally assessed and showed an improved performance during both the transient and steady state operation conditions. However, the proposed system has a stability issue related to the rate of change of the observer gains. In addition, the effect of sudden load disturbance on the adaptation mechanism performance was not tested.

Gadoue *et al.* [42] proposed a sliding mode adaptation mechanism to replace the PI controller in the conventional MRAS observer. Lyapunov theory is used to derive the sliding mode-based adaptation mechanism to ensure the sytem stability. The performance of the proposed and the conventional methods is compared in open-loop and closed loop (sensorless) modes of operation. Although this scheme is shown to improve the estimator dynamic response, it causes a considerable amount of chattering in the estimated speed signal, and a low-pass filter is needed to smooth out the estimated rotor speed. Another solution was also proposed in [42] where the PI controller is replaced by a fuzzy logic (FL)-based adaption mechanism. This scheme shows improvement in the estimator dynamic response, but the computational complexity of the FL controller is the main drawback of this scheme.

2.5 Predictive Control for Sensorless Control of Electric Drives

General model predictive controllers (MPC) can be classified into classical MPC and finite control set-model predictive controllers (FCS-MPC) [78]. In classical MPC, the controller generates a continuous voltage vector and a modulator is used to apply this voltage to the inverter. In FCS-MPC the controller directly produces a switching state of the inverter [79].

FCS-MPC has increasingly gained popularity and has been applied in many different applications because of its simplicity, compact design and flexibility to include any performance specifications [10, 13, 14, 80-84]. For example in [84] , an FSC-MPC was applied to drive an IM fed by a matrix converter to increase the system efficiency, and in [83] current control of a five-phase IM is applied based on the FCS-MPC control principle.

Over the last few years, interest has grown in the use of predictive control techniques with sensorless applications. In [17-20], predictive control is applied to permanent-magnet sensorless motor drives, and in [23] a predictive torque control with sliding mode feedback is used with a sensorless IM drive.

A speed-sensorless control system for an IM with a predictive current controller has been proposed in [24], where it has been claimed that this combination can make the system simple, robust and able to operate at very high and extremely low speeds. However, the system performance with the predictive current controller was not compared with the classical system which uses PI current controllers.

A new generalized predictive controller has been introduced in [25] for the control of sensorless IM drives. The controller structure allows controlling of the speed and rotor flux, taking into consideration the stator windings voltage and current constraints. The proposed controller is applied with the MRAS speed estimator. The result shows faster dynamic response in comparison with other speed controllers. However the effect of the proposed controller on the accuracy of the speed estimation is not discussed.

In [26], a predictive torque controller (PTC) was applied with rotor flux- based MRAS observer to develop a sensorless PTC system for IM. A flux compensator was used to improve the reference model of the observer. The results show an improved dynamic of the whole system. However, the system did not perform well when the speed crossed the zero point during the speed reversal experiment. This was due to the low switching frequency of the power switches which affects the PTC operation.

In [31] a sensorless control of IM drive using a finite set PTC (FS-PTC) is applied. The prediction accuracy was improved by using an EKF to estimate the rotor speed and stator currents. The estimated stator currents were used to avoid the measurement noise. The result shows a reduction in the stator current total harmonic distortion (THD) in the stator currents and improved flux and speed estimation.

From all the aforementioned publications, it can be concluded that using predictive controllers can improve the drive system performance and increase the system robustness to motor parameters variation. However, the prediction principle so far was only applied on the controller side of the drive, and none of the examples considered above has introduced the prediction principle into the design of the speed estimator itself. In the following chapters a predictive sensorless speed observer is introduced to provide robust and accurate speed estimation.

2.6 Summary

In this chapter a review of the different schemes used for speed estimation in the IM drives has been provided. This review has focused on the MRAS-based strategies and provided a description of the different problems affecting the performance of these observers at low speed operation. Various methods employed to improve the design of the adaptation mechanism have been also carefully discussed. It appears that, despite all the attempts to improve the adaptation mechanism design, finding a design that can operates at the same efficiency level at different speeds and operating conditions and which is less affected by the machine parameters variation is still challenging and needs more investigation.

The way the proposed predictive MRAS observer operates make it more suitable for the applications where accurate and quick position estimation is required. For example this estimator is very suitable for conveyor belt applications where accurate positioning is required for processes such as assembly, inspection, and feeding into equipment. In such applications, the precise product movement is critical.

CHAPTER 3

Model Reference Adaptive Systems for Vector Control Induction Motor Drive

3.1 Introduction

Induction motors (IM), due to their simplicity, rigid structure and low cost, are used widely in many applications. By using advanced control strategies like direct torque control (DTC) and field oriented control (FOC), IM drives have become more dominant in many applications especially those requiring high performance operation. The fast development of microprocessors and power electronics has also played a key role in the IM drives development, as it allows the use of control techniques that require high computational effort. However the design of such control systems requires an accurate model of the induction machine. This chapter introduces the induction machine dynamic modelling using space vector theory. The machine model is used then to explain the principle of vector control. Finally, the two-axis machine model is used to describe the MRAS speed observer principle and to formulate the equations of the observer mathematical model.

3.2 The Dynamic Model of the Induction Machine

Generally, the equivalent circuit of the IM is used to analyse the performance of the machine and to calculate its different quantities at different loading conditions. However, this representation is limited and cannot be used to study the transient operation of the machine or even the steady state operation when the machine is fed by a non-sinusoidal source, such as when the machine is fed by a power electronic converter. Based on the two-axis theory of the electrical machines and the space vector theory, a dynamic model of the IM can be derived to allow the analysis of the machine performance at any operating condition and any type of power supply.

For three phase IMs, the stator current space vector in the stationary reference frame can be expressed as [2]:

$$\bar{i}_s = \frac{2}{3}(i_{sA} + a i_{sB} + a^2 i_{sC}) \tag{3.1}$$

where:

$$a = e^{j\frac{2\pi}{3}} = -\frac{1}{2} + j\frac{\sqrt{2}}{3}$$

$$a^2 = e^{j\frac{4\pi}{3}} = -\frac{1}{2} - j\frac{\sqrt{2}}{3} \tag{3.2}$$

i_{sA}, i_{sB} and i_{sC} are the stator three-phase currents which are sinusoidal and shifted by 120º in the time domain. These currents flow in the stator three phase windings, which are spatially displaced by 120º, to produce a rotating magnetic flux.

To find the two axis components of the stator current vector $\bar{i}_s$, (3.2) is substituted in (3.1), and as a result, the real part of the stator current will represent the *D* axis current component, whereas the imaginary part will be the *Q* axis current component. This leads to the following equations:

$$i_{sD} = \frac{2}{3}i_{sA} - \frac{1}{3}i_{sB} - \frac{1}{3}i_{sC}$$

$$i_{sQ} = \frac{1}{\sqrt{3}}i_{sB} - \frac{1}{\sqrt{3}}i_{sC} \tag{3.3}$$

where i_{sD} and i_{sQ} are the two-phase equivalent currents that flow in the virtual two phase windings s_D and s_Q which are spatially displaced by 90º [2]. The stator current vector $\bar{i}_s$ can be accordingly written as:

$$\bar{i}_s = |\bar{i}_s| e^{j\alpha_s} = i_{sD} + j i_{sQ} \tag{3.4}$$

where α_s is the angle between the current vector and the s_D axis.

From (3.1) and (3.4), the transformation from three-phase to two-phase can be written in matrix form as:

$$\begin{bmatrix} i_{sD} \\ i_{sQ} \end{bmatrix} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}\begin{bmatrix} i_{sA} \\ i_{sB} \\ i_{sC} \end{bmatrix} \tag{3.5}$$

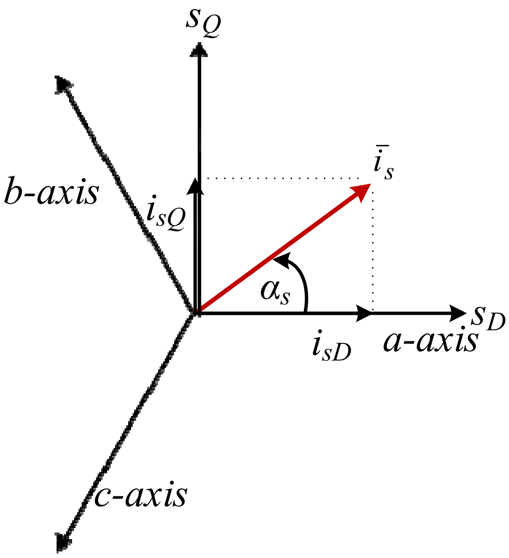


Fig. 3.1 The stator current space vector

Similarly, the two to three phase transformations can be written as:

$$i_{sA} = i_{sD}$$

$$i_{sB} = -\frac{1}{2}i_{sD} + \frac{\sqrt{3}}{2}i_{sQ}$$

$$i_{sC} = -\frac{1}{2}i_{sD} - \frac{\sqrt{3}}{2}i_{sQ} \tag{3.6}$$

and in matrix form the transformation can be expressed as:

$$\begin{bmatrix} i_{sA} \\ i_{sB} \\ i_{sC} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}\begin{bmatrix} i_{sD} \\ i_{sQ} \end{bmatrix} \tag{3.7}$$

To apply FOC, or as it is also called vector control (VC), another frame transformation is required. This transformation is between the stationary frames *(D-Q)* and the excitation (synchronous) frames *(d-q)* which are shown in Fig. 3.2. The stationary frame is fixed to the machine stator, whereas the excitation frame rotates at the synchronous speed.

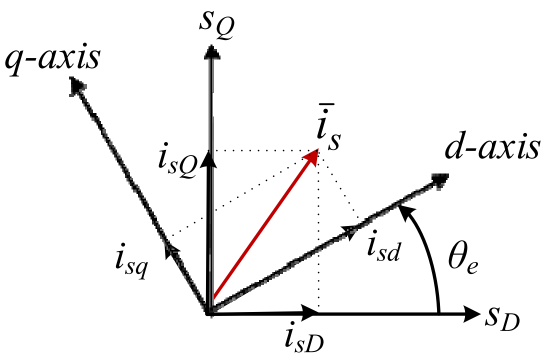


Fig. 3.2 The transformation from the stationary to the synchronous frame

The equation that describes the transformation from the stationary frame to the synchronous frame can be written as:

$$\bar{i}_s^e = i_{sd} + j \cdot i_{sq} = \bar{i}_s^s e^{-j\theta_e} \;\; = (i_{sD} + ji_{sQ})e^{-j\theta_e} \tag{3.8}$$

The transformation can also be expressed in the matrix form as:

$$\begin{bmatrix} i_{sd} \\ i_{sq} \end{bmatrix} = \begin{bmatrix} \cos\theta_e & \sin\theta_e \\ -\sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} i_{sD} \\ i_{sQ} \end{bmatrix} \tag{3.9}$$

The transformation from the excitation frame to the stationary frame can be similarly written as:

$$\bar{i}_s^s = i_{sD} + j \cdot i_{sQ} = \bar{i}_s^e e^{j\theta_e} = (i_{sd} + ji_{sq})e^{j\theta_e} \tag{3.10}$$

and in matrix form:

$$\begin{bmatrix} i_{sD} \\ i_{sQ} \end{bmatrix} = \begin{bmatrix} \cos\theta_e & -\sin\theta_e \\ \sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} i_{sd} \\ i_{sq} \end{bmatrix} \tag{3.11}$$

Similarly, the flux and stator space vector voltage in the stator reference frame can be defined as:

$$\bar{v}_s = \frac{2}{3}(v_{sA} + a v_{sB} + a^2 v_{sC}) \tag{3.12}$$

$$\bar{v}_s = v_{sD} + j v_{sQ} \tag{3.13}$$

$$\bar{\psi}_s = \frac{2}{3}(\psi_{sA} + a\psi_{sB} + a^2\psi_{sC}) \tag{3.14}$$

The same procedure applied to find the stator current space vector in a stator reference frame, can be used to find the rotor current space vector in a reference frame fixed to the machine rotor:

$$\bar{i}_r = \frac{2}{3}(i_{ra} + a i_{rb} + a^2 i_{rc}) \tag{3.15}$$

$$\bar{i}_r = |\bar{i}_r| e^{j\alpha_r} = i_{r\alpha} + j i_{r\beta} \tag{3.16}$$

where $i_{r\alpha}$ and $i_{r\beta}$ are the two phase equivalent currents that flow in the virtual two-phase rotor windings r_α and r_β which are spatially displaced by 90º, and θ_r is the rotor position angle [2].

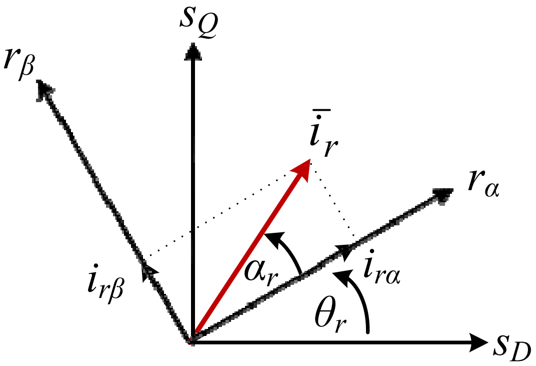


Fig. 3.3 The rotor current space vector

The IM mathematical model can be described by a set of equations which are based on Kirchhoff's laws. The detail of the IM modelling is stated in Appendix A. By applying Kirchhoff's laws on the stator and rotor windings the following equations are obtained:

$$\bar{v}_s^s = R_s \bar{i}_s^s + \frac{d\bar{\psi}_s^s}{dt} \tag{3.17}$$

$$\bar{v}_r^r = R_r \bar{i}_r^r + \frac{d\bar{\psi}_r^r}{dt} \tag{3.18}$$

The subscripts r and s stand for the rotor and stator quantities respectively, and the superscripts r and s stand for the stator and stator reference frames.

(3.17) and (3.18) can be also described in terms of the vectors coordinates in the stationary reference frame as:

$$v_{sD} = R_s i_{sD} + \frac{d\psi_{sD}}{dt}$$

$$v_{sQ} = R_s i_{sQ} + \frac{d\psi_{sQ}}{dt} \tag{3.19}$$

$$v_{rd} = R_r i_{rd} + \frac{d\psi_{rd}}{dt} + \omega_r \psi_{rq}$$

$$v_{rq} = R_r i_{rq} + \frac{d\psi_{rq}}{dt} - \omega_r \psi_{rd} \tag{3.20}$$

The stator and rotor flux linkages can be expressed as:

$$\psi_{sD} = L_s i_{sD} + L_m i_{rd}$$

$$\psi_{sQ} = L_s i_{sQ} + L_m i_{rq} \tag{3.21}$$

$$\psi_{rd} = L_m i_{sD} + L_r i_{rd}$$

$$\psi_{Rq} = L_m i_{sQ} + L_r i_{rq} \tag{3.22}$$

By solving (3.19) to (3.22), new equations that describe the IM dynamic can be found in terms of the stator and the rotor current vectors represented in the stationary reference frame:

$$v_{sD} = (R_s + L_s p)i_{sD} + L_m p i_{rd}$$

$$v_{sQ} = (R_s + L_s p)i_{sQ} + L_m p i_{rq} \tag{3.23}$$

$$v_{rd} = (R_r + L_r p)i_{sD} + L_m p i_{sD} + \omega_r (L_m i_{sQ} + L_r i_{rq})$$

$$v_{rq} = (R_r + L_r p)i_{sQ} + L_m p i_{sQ} - \omega_r (L_m i_{sD} + L_r i_{rd}) \tag{3.24}$$

where p denotes the differential operator.

Using space vector notation, equations (3.23) and (3.24) can be written in the matrix form as:

$$\begin{bmatrix} \bar{v}_s^s \\ \bar{v}_r^s \end{bmatrix} = \begin{bmatrix} R_s + L_s p & L_m p \\ L_m(p - j\omega_r) & R_r + L_r(p - j\omega_r) \end{bmatrix} \begin{bmatrix} \bar{i}_s^s \\ \bar{i}_r^s \end{bmatrix} \tag{3.25}$$

A convenient way to understand and design the vector control system is to rewrite the rotor and stator voltage equations in terms of stator current and rotor flux, therefore these equations for a squirrel-cage IM can be written as:

$$v_{sD} = R_s i_{sD} + \sigma L_s p i_{sD} + \frac{L_m}{L_r} p \psi_{rd}$$

$$v_{sQ} = R_s i_{sQ} + \sigma L_s p i_{sQ} + \frac{L_m}{L_r} p \psi_{rq} \tag{3.26}$$

$$0 = \frac{1}{T_r}(\psi_{rd} - L_m i_{sD}) + p\psi_{rd} + \omega_r \psi_{rq}$$

$$0 = \frac{1}{T_r}(\psi_{rq} - L_m i_{sQ}) + p\psi_{rq} - \omega_r \psi_{rd} \tag{3.27}$$

where σ is called the leakage coefficient and it can be found from:

$$\sigma = 1 - \frac{L_m^2}{L_s L_r} \tag{3.28}$$

T_r is the rotor time constant and it is describes as:

$$T_r = \frac{L_r}{R_r} \tag{3.29}$$

From the previous equations, the state space representation of the squirrel cage IM can be derived. The states of the state space model are stator currents and the rotor flux linkages.

$$\begin{bmatrix} pi_{sD} \\ pi_{sQ} \\ p\psi_{rd} \\ p\psi_{rq} \end{bmatrix} = \begin{bmatrix} -a_1 & 0 & a_2 & a_3\omega_r \\ 0 & -a_1 & -a_3\omega_r & a_2 \\ \frac{L_m}{T_r} & 0 & -\frac{1}{T_r} & -\omega_r \\ 0 & \frac{L_m}{T_r} & \omega_r & -\frac{1}{T_r} \end{bmatrix} \begin{bmatrix} i_{sD} \\ i_{sQ} \\ \psi_{rd} \\ \psi_{rq} \end{bmatrix} + \begin{bmatrix} \frac{1}{\sigma L_s} & 0 \\ 0 & \frac{1}{\sigma L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_{sD} \\ v_{sQ} \end{bmatrix} \tag{3.30}$$

where:

$$a_1 = \frac{R_s}{\sigma L_s} + \frac{1-\sigma}{\sigma T_r} \tag{3.31}$$

$$a_2 = \frac{L_m}{\sigma L_s L_r T_r} \tag{3.32}$$

$$a_3 = \frac{L_m}{\sigma L_s L_r} \tag{3.33}$$

In comparison with the standard form of the state space model which is:

$$\dot{\boldsymbol{X}}(t) = \boldsymbol{A}\boldsymbol{X}(t) + \boldsymbol{B}\boldsymbol{U}(t)$$

$$\boldsymbol{Y}(t) = \boldsymbol{C}\boldsymbol{X}(t) + \boldsymbol{D}\boldsymbol{U}(t) \tag{3.34}$$

The following can be written:

$$\mathbf{X}(t) = [i_{sD} \quad i_{sQ} \quad \psi_{rd} \quad \psi_{rq}]^T \tag{3.35}$$

$$\mathbf{U}(t) = \begin{bmatrix} v_{sD} & v_{sQ} \end{bmatrix}^T, \tag{3.36}$$

$$\mathbf{Y}(t) = \begin{bmatrix} i_{sD} & i_{sQ} \end{bmatrix}^T \tag{3.37}$$

$$\boldsymbol{A} = \begin{bmatrix} -a_1 & 0 & a_2 & a_3\omega_r \\ 0 & -a_1 & -a_3\omega_r & a_2 \\ \frac{L_m}{T_r} & 0 & -\frac{1}{T_r} & -\omega_r \\ 0 & \frac{L_m}{T_r} & \omega_r & -\frac{1}{T_r} \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} \frac{1}{\sigma L_s} & 0 \\ 0 & \frac{1}{\sigma L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.38}$$

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \boldsymbol{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.39}$$

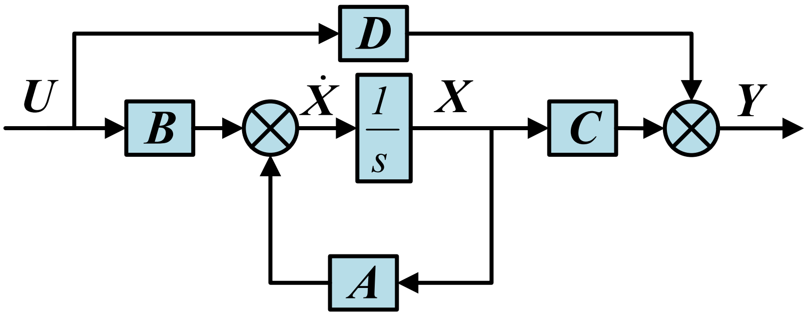


Fig. 3.4 The state space model

Finally, the electromagnetic Torque can be found from the following equation (A.88):

$$T_e = \frac{3}{2} P \frac{L_m}{L_r} (\psi_{rd} i_{sQ} - \psi_{rq} i_{sD}) \tag{r3.40}$$

where P is the number of pole pairs.

3.3 Vector Control

By choosing the *d-axis* of the synchronous reference frame to be aligned with the rotor flux space vector, the rotor flux can be controlled separately by adjusting the *d* component of the stator current and the torque can be controlled by changing the *q* component of the stator current. This makes the control of the IM similar to that of the separately excited DC machines. Vector controlled is sometimes referred to as Field Oriented Control (FOC), and Fig. 3.5 shows phasor diagram of a vector control system.

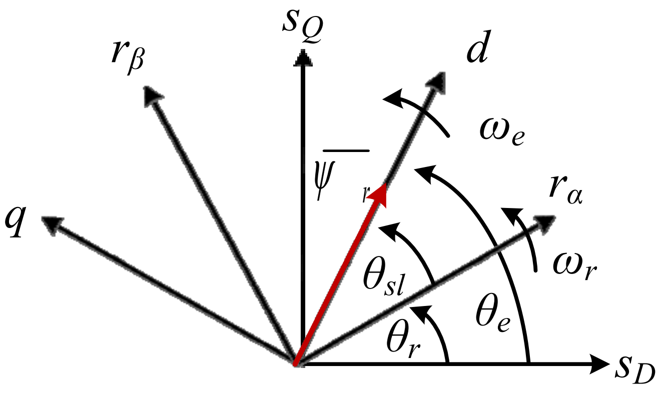


Fig. 3.5 Vector control principle

To find the equations that explain vector control principle, the model of the squirrel cage IM in a synchronous reference frame are considered. The voltage equations for both the stator and the rotor are written as:

$$v_{sd} = R_s i_{sd} + p\psi_{sd} - \omega_e \psi_{sq}$$

$$v_{sq} = R_s i_{sq} + p\psi_{sq} + \omega_e \psi_{sd} \tag{3.41}$$

$$0 = R_r i_{rd} + p\psi_{rd} - \omega_{sl} \psi_{rq}$$

$$0 = R_r i_{rq} + p\psi_{rq} + \omega_{sl} \psi_{rd} \tag{3.42}$$

However, the rotor flux can be expressed in terms of stator and rotor currents as:

$$\psi_{rd} = L_m i_{sd} + L_r i_{rd}$$

$$\psi_{rq} = L_m i_{sq} + L_r i_{rq} \tag{3.43}$$

Because in FOC the *d-axis* of the frame is aligned with the rotor flux space vector, it can be written:

$$\overline{\psi}_r = \psi_{rd} \text{ , } \psi_{rq} = 0 \text{ and } p\psi_{rq} = 0 \tag{3.44}$$

By solving (3.41) to (3.44):

$$R_r i_{rd} + p\psi_{rd} = 0 \tag{3.45}$$

$$R_r i_{rq} + \omega_{sl}\psi_{rd} = 0 \tag{3.46}$$

$$L_m i_{sq} + L_r i_{rq} = 0 \tag{3.47}$$

Rearranging (3.47) gives:

$$i_{rq} = \frac{L_m}{L_r} i_{sq} \tag{3.48}$$

The slip rotating speed can then be found by substituting (3.48) into (3.46):

$$\omega_{sl} = \frac{L_m}{T_r \psi_{rd}} i_{sq} \tag{3.49}$$

At steady state $p\psi_{rq} = 0$, therefore from (3.45) it can be said that $i_{rd} = 0$ at steady state, and as a result (3.43) can be written as:

$$\psi_{rd} = L_m i_{sd} \tag{3.50}$$

Substituting (3.50) into (3.49) the slip angular frequency ω_{sl} can be written as:

$$\omega_{sl} = \omega_e - \omega_r = \frac{1}{T_r}\frac{i_{sq}}{i_{sd}} \tag{3.51}$$

Practically, the speed angular frequency is found from the reference currents components instead of the actual currents [2] which means (3.49) can be rewritten as:

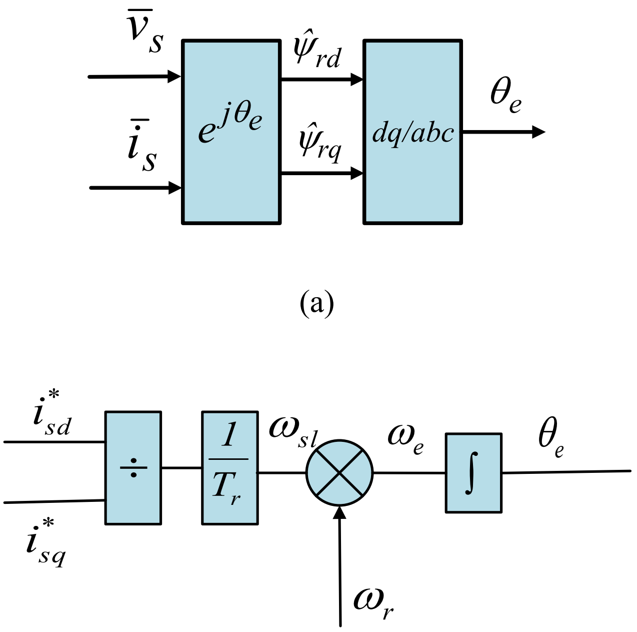
$$\omega_{sl} = \frac{1}{T_r}\frac{i_{sq}^*}{i_{sd}^*} \tag{3.52}$$

and hence the rotor flux position can be found from:

$$\theta_e = \theta_r + \int \frac{1}{T_r}\frac{i_{sq}^*}{i_{sd}^*}dt \tag{3.53}$$

The electromagnetic torque can be found by substituting (3.44) into (3.41) which gives:

$$T_e = \frac{3}{2}P\frac{L_m}{L_r}\psi_{rd}i_{sq} \tag{3.54}$$



(a)

(b)

Fig. 3.6 Rotor flux Position (a) Direct orientation (b) Indirect orientation

If a constant K_t is defined as:

$$K_t = \frac{3}{2} P \frac{L_m}{L_r} \tag{3.55}$$

then, the torque equation can be rewritten as:

$$T_e = K_t \psi_{rd} i_{sq} \tag{3.56}$$

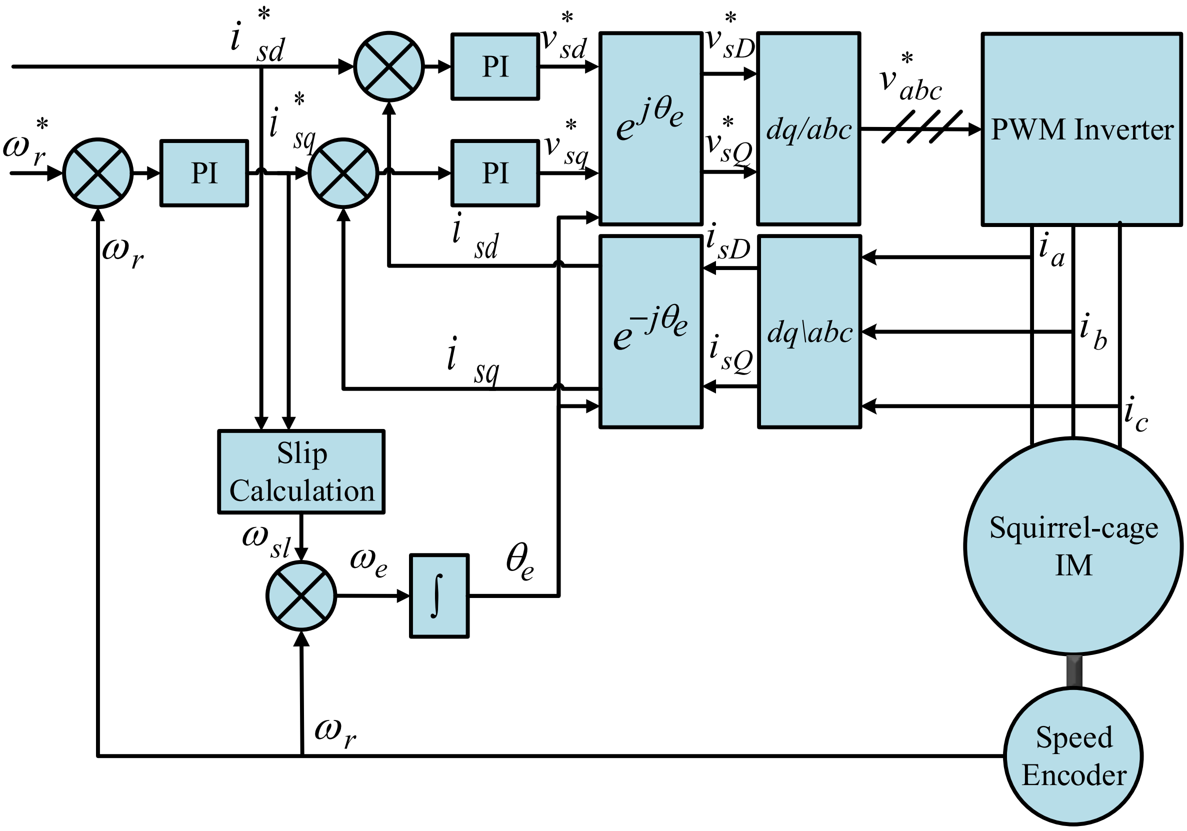


Fig. 3.7 Indirect vector control of IM

Obviously from (3.50) and (3.56), applying vector control, makes the dynamic model of the IM similar to that of the DC machine, with i_{sd} and i_{sq} as analogues to the field and armature currents, respectively. Vector control also allows control of the different AC variables of the IMs as DC quantities since the control is performed in a synchronisingly rotating reference frame, and all the variables appear constant at the steady state.

As it has been discussed above, in order to implement vector control, the accurate position of the rotor flux θ_e is required. This angle can be found by applying two different methods which leads to two types of vector control; direct vector control, and indirect vector control.

In the direct vector control, the rotor flux space vector position is measured directly, which can be found by either using a Hall-effect magnetic field sensor, or by applying a flux observer which use the measured stator currents and voltages to estimate the rotor flux position, whereas, the indirect vector control uses the machine model to find the slip speed, and the rotor position is then added to get the rotor flux position. Clearly, applying this method is simpler because it does not require a flux sensor or observer, but it is sensitive to the rotor time constant variation which can cause the system performance to deteriorate.

3.4 Model Reference Adaptive Systems (MRAS) for Induction Machines

Any MRAS system for parameter identification consists basically of three different components: the reference model which defines the desired states X_R, the adaptive model which produces the estimated value of the same states $\hat{X}_A$ and the adaption mechanism which, depending on the error between the reference and adaptive model, generates an estimation of the parameter to be estimated$\hat{\theta}$. Afterward, $\hat{\theta}$ is used to adjust the adaptive model. The proper design of the adaption mechanism ensures that the error between the two models converge to zero after repeating the described process a number of times [2].

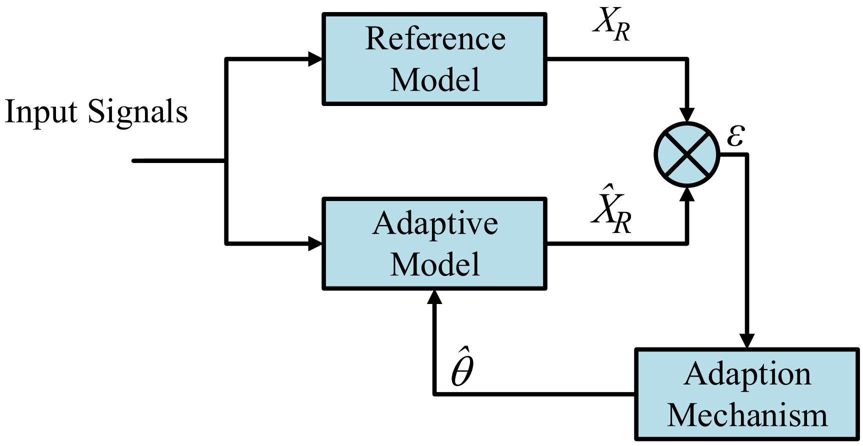


Fig. 3.8 Block diagram of a basic MRAS observer

The modelling of the rotor flux linkage MRAS speed estimator (observer) will discussed in the next section.

3.4.1 Rotor flux MRAS speed observer

In order to design this observer, two models should be found that can calculate the same outputs. The first model (the reference model) should be able to perform the calculation without the need for information on the estimated parameter, which is the speed in this case, and the second model (the adaptive model) does depend on the speed in the calculation. The rotor flux components are chosen to be the outputs of both models and therefore this scheme is called rotor flux MRAS observer.

The stator voltage equations in a stationary reference frame in terms of stator currents and rotor flux linkage can be written as:

$$v_{sD} = R_s i_{sD} + \sigma L_s p i_{sD} + \frac{L_m}{L_r} p \psi_{rd}$$

$$v_{sQ} = R_s i_{sQ} + \sigma L_s p i_{sQ} + \frac{L_m}{L_r} p \psi_{rq} \tag{3.57}$$

Likewise, the rotor voltage equations in a stationary reference frame in terms of stator currents and rotor flux linkage can be written as:

$$0 = \frac{1}{T_r}\psi_{rd} - \frac{L_m}{T_r} i_{sD} + p\psi_{rd} + \omega_r \psi_{rq}$$

$$0 = \frac{1}{T_r}\psi_{rq} - \frac{L_m}{T_r} i_{sQ} + p\psi_{rq} - \omega_r \psi_{rd} \tag{3.58}$$

Rearranging (3.57) leads to:

$$p\psi_{rd} = \frac{L_r}{L_m}(v_{sD} - R_s i_{sD} - \sigma L_s p i_{sD})$$

$$p\psi_{rq} = \frac{L_r}{L_m}(v_{sQ} - R_s i_{sQ} - \sigma L_s p i_{sQ}) \tag{3.59}$$

and from (3.58) it can be written:

$$p\hat{\psi}_{rd} = \frac{L_m}{T_r} i_{sD} - \frac{1}{T_r}\hat{\psi}_{rd} - \hat{\omega}_r \hat{\psi}_{rq}$$

$$p\hat{\psi}_{rq} = \frac{L_m}{T_r} i_{sQ} - \frac{1}{T_r}\hat{\psi}_{rq} + \hat{\omega}_r \hat{\psi}_{rd} \qquad (3.60)$$

From (3.59) and (3.60) it can be noticed that the rotor flux components are the common outputs of the two sets of equations, where (3.59) can calculate the rotor flux components without the need for the rotor speed information and therefore it can be used as a reference model. This model is sometimes referred to as the Voltage Model (VM). Likewise, (3.60) has a speed dependant term and consequently it can be considered as an Adaptive Model which calculates the flux components based on the speed value, and this model is referred to as the Current Model (CM).

The first rotor flux-based MRAS speed observer was introduced in [46], where (3.59) and (3.60) were used as reference and adaptive models respectively. With a proper adaption mechanism, the system was able to estimate the rotor speed by minimizing the generated error between the two models. The diagram shown in Fig. 3.9 illustrates the general structure of the rotor flux-based MRAS scheme. The adaption mechanism design will be described separately in the following section.

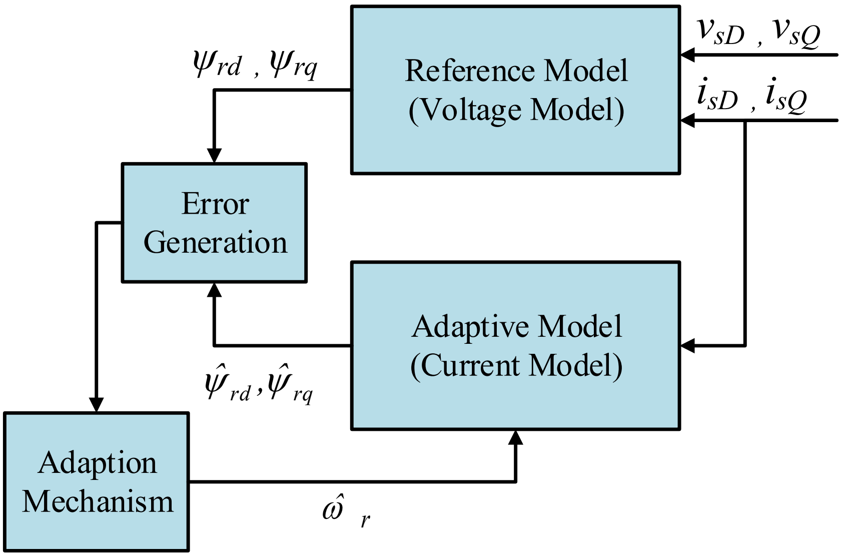


Fig. 3.9 Rotor flux linkage MRAS speed observer

From (3.60), it can be noticed that in the adaptive model equations, there is a presence of a cross coupling components between the two equations, and these components are speed dependant, which means that they can lead to an instability problem, as mentioned in [85]. To address this problem, a rotor reference frame representation of the rotor flux can be used instead to calculate the rotor flux linkage components [68, 85]. In this representation, the flux vector components can be rewritten as:

$$\overline{\psi}_r^r = L_m \bar{i}_s^r + L_r \bar{i}_r^r \tag{3.61}$$

The rotor voltage equation of a squirrel cage IM represented in the rotor reference frame is:

$$p\overline{\psi}_r^r = -R_r \bar{i}_r^r \tag{3.62}$$

From (3.61) and (3.62), the rotor flux linkage space vector in the rotor reference frame can be expressed as:

$$p\overline{\psi}_r^r = -\frac{1}{T_r}\overline{\psi}_r^r + \frac{L_m}{T_r}\bar{i}_s^r \tag{3.63}$$

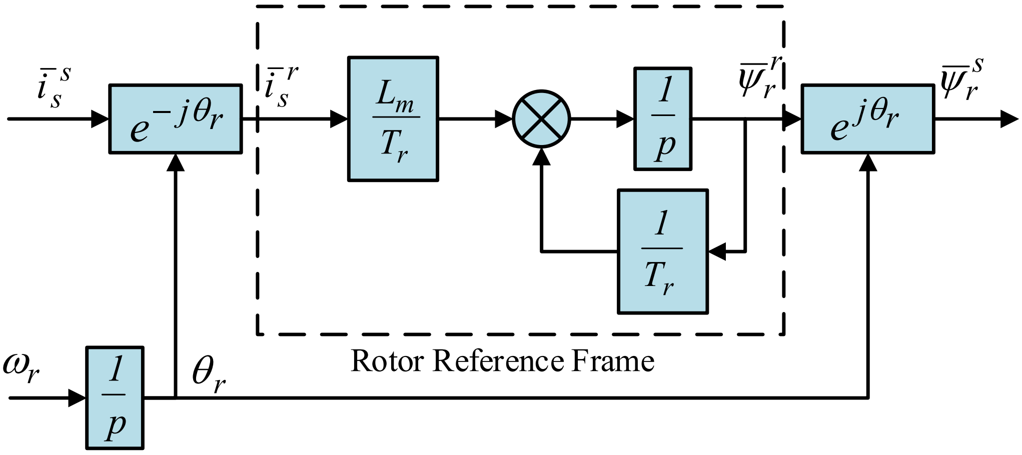


Fig. 3.10 The rotor reference frame implementation of the adaptive model

From (3.63) it can be noticed that, the stator current vector represented in the rotor reference frame is needed in order to calculate the rotor flux linkage represented in the same reference frames, which can be found by applying:

$$\bar{i}_s^r = \bar{i}_s^s e^{-j\theta_r} \tag{3.64}$$

As the rotor flux represented in the stationary reference frame is required at the output of the adaptive model, the following transformation can be applied :

$$\overline{\psi}_r^s = \overline{\psi}_r^r e^{j\theta_r} \tag{3.65}$$

3.4.2 The adaption mechanism design

As mentioned previously, in the rotor flux-based MRAS observers, the speed is estimated by calculating the difference between the flux components at the output of the reference and the adaptive models, which is also referred to as the estimation error. The estimation error is then processed within an adaption mechanism to estimate the rotor speed, which is then used to adjust the adaptive model and hence to reduce the error between the outputs of the two models. This procedure will continue until the error between the two model outputs converge to zero.

Within the literature, the Popov's hyperstability theory has been mainly used to design the adaption mechanism of MRAS observers [2, 46]. The details of hyperstability theory are beyond the scope of this thesis. However, a brief description is given here to help understanding of the design of the adaptation mechanism applied in this thesis.

According to the hyperstability theory, in order for the stability analysis to be carried out, the feedback system needs to be presented as linear and nonlinear feed forward and feedback subsystems respectively [2], Fig 3.11, the system will be considered to be stable, if the feedforward subsystem transfer function is strictly real and positive and the nonlinear feedback subsystem satisfies: $\int_0^t \varepsilon^T \boldsymbol{W} dt \geq -\gamma_0^2$ where γ_0^2 is an arbitrary positive constant [46, 55]. In Fig. 3.11, ***U*** is the input of the linear subsystem, ***ε*** is its output, ***W*** is the output of the nonlinear subsystem and ***U=-W***.

First of all, in order to find the equivalent linear and nonlinear subsystems of the rotor flux MRAS observer, the error vector ***ε,*** which is the difference between the reference and adaptive fluxes, needs to be defined as:

$$\varepsilon_d = \psi_{rd} - \hat{\psi}_{rd}$$

$$\varepsilon_q = \psi_{rq} - \hat{\psi}_{rq} \tag{3.66}$$

Which can be written as an error vector as:

$$\varepsilon = \begin{bmatrix} \varepsilon_d & \varepsilon_q \end{bmatrix}^T \tag{3.67}$$

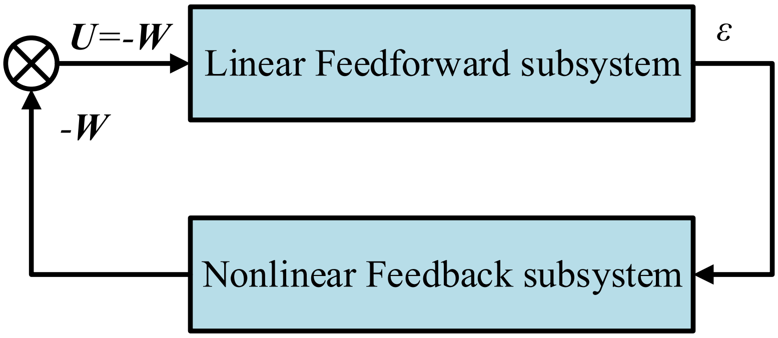


Fig. 3.11 The hyperstability theory equivalent system block diagram

By differentiating (3.66) and substituting in (3.60):

$$p\varepsilon_d = -\frac{1}{T_r}\varepsilon_d - \omega_r\varepsilon_q - (\omega_r - \hat{\omega}_r)\hat{\psi}_{rq}$$

$$p\varepsilon_q = -\frac{1}{T_r}\varepsilon_q + \omega_r\varepsilon_d + (\omega_r - \hat{\omega}_r)\hat{\psi}_{rd} \tag{3.68}$$

And in matrix form:

$$\begin{bmatrix} p\varepsilon_d \\ p\varepsilon_q \end{bmatrix} = \begin{bmatrix} -\frac{1}{T_r} & -\omega_r \\ \omega_r & -\frac{1}{T_r} \end{bmatrix} \begin{bmatrix} \varepsilon_d \\ \varepsilon_q \end{bmatrix} + \begin{bmatrix} -\hat{\psi}_{rq} \\ \hat{\psi}_{rd} \end{bmatrix} (\omega_r - \hat{\omega}_r) \tag{3.69}$$

If the following form is considered:

$$A = \begin{bmatrix} -\frac{1}{T_r} & -\omega_r \\ \omega_r & -\frac{1}{T_r} \end{bmatrix}, \mathbf{W} = \begin{bmatrix} \hat{\psi}_{rq} \\ -\hat{\psi}_{rd} \end{bmatrix} (\omega_r - \hat{\omega}_r) \tag{3.70}$$

The system can be presented as:

$$p\varepsilon = A\varepsilon - W \tag{3.71}$$

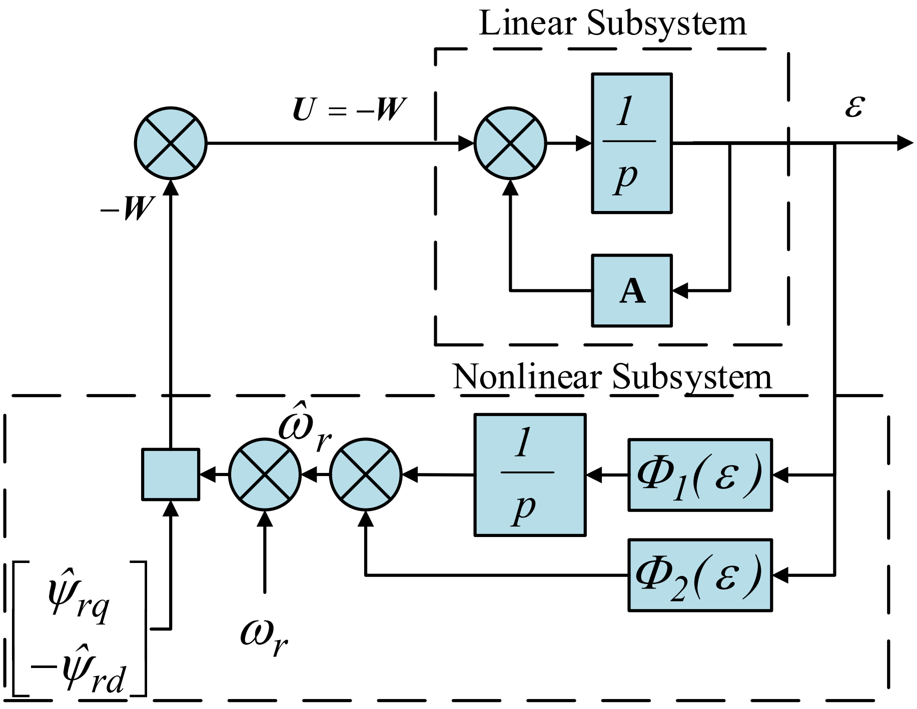


Fig. 3.12 The linear and nonlinear system presentation

Equation (3.71) is similar to the linear and nonlinear equivalent subsystems in Fig. 3.12 and the transfer function of the linear system is obviously positive and real.

If the adaption law is defined as:

$$\hat{\omega}_r = \Phi_2(\varepsilon) + \int_0^t \Phi_1(\varepsilon) d\tau \tag{3.72}$$

By substituting (3.72), (3.70) and (3.67) in $\int_0^t \varepsilon^T \boldsymbol{W} dt \geq -\gamma_0^2$ the following can be found:

$$\int_0^t(\varepsilon_d\hat{\psi}_{rq}-\varepsilon_q\hat{\psi}_{rd})(\omega_r-\Phi_2(\varepsilon)-\int_0^t\Phi_1(\varepsilon)d\tau)dt\geq-\gamma_0^2 \tag{3.73}$$

This inequality can be satisfied if the following functions are used [46]:

$$\Phi_1(\varepsilon)=k_2(\varepsilon_q\hat{\psi}_{rd}-\varepsilon_d\hat{\psi}_{rq})=K_i(\psi_{rq}\hat{\psi}_{rd}-\psi_{rd}\hat{\psi}_{rq})$$

$$\Phi_2(\varepsilon)=k_1(\varepsilon_q\hat{\psi}_{rd}-\varepsilon_d\hat{\psi}_{rq})=K_p(\psi_{rq}\hat{\psi}_{rd}-\psi_{rd}\hat{\psi}_{rq}) \tag{3.74}$$

If a speed tuning signal is defined as:

$$\varepsilon_\omega=\psi_{rq}\hat{\psi}_{rd}-\psi_{rd}\hat{\psi}_{rq} \tag{3.75}$$

From (3.74) ε_ω can be minimized by applying a *PI* controller which at the same time can generate the estimated speed of the machine rotor. As a conclusion, the IM estimated speed can be expressed as:

$$\hat{\omega}_r=(K_p+\frac{K_i}{p})\varepsilon_\omega \tag{3.76}$$

Fig. 3.13 shows the adaption mechanism of the rotor flux linkage MRAS observer.

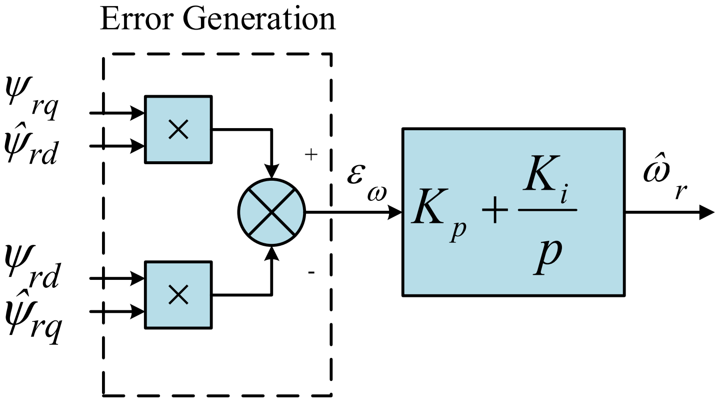


Fig. 3.13 The adaption mechanism of the rotor flux linkage MRAS observer

3.4.3 The system representation in discrete time

Until now, all the equations derived are represented in continues time domain. However, because of the fact that the practical implementation of the drive system requires these equations to be processed digitally, the discrete representation of the system is considered within this section. To perform the continues to discrete-time transformation, all the differential equations need to be transferred into Laplace domain. The resultant equations can then be transferred into z-domain by applying Euler backward transformation.

3.4.3.1 Discrete-time reference model

The continues-time equations that represent the reference model are:

$$p\psi_{rd} = \frac{L_r}{L_m}(v_{sD} - R_s i_{sD} - \sigma L_s p i_{sD})$$

$$p\psi_{rq} = \frac{L_r}{L_m}(v_{sQ} - R_s i_{sQ} - \sigma L_s p i_{sQ}) \qquad (3.77)$$

Applying Laplace transformation on (3.77) yields:

$$s\psi_{rd}(s) = \frac{L_r}{L_m}\left[v_{sD}(s) - R_s i_{sD}(s) - \sigma L_s s i_{sD}(s)\right]$$

$$s\psi_{rq}(s) = \frac{L_r}{L_m}\left[v_{sQ}(s) - R_s i_{sQ}(s) - \sigma L_s s i_{sQ}(s)\right] \qquad (3.78)$$

According to Euler backward transformation, every *s* can be replaced by $s = \frac{1 - z^{-1}}{Ts}$ and (3.78) becomes:

$$\left[\frac{1 - z^{-1}}{Ts}\right]\psi_{rd}[k] = \frac{L_r}{L_m}\left[v_{sD}[k] - R_s i_{sD}[k] - \sigma L_s \left[\frac{1 - z^{-1}}{Ts}\right] i_{sD}[k]\right]$$

$$\left[\frac{1-z^{-1}}{Ts}\right]\psi_{rq}[k]=\frac{L_r}{L_m}\left[v_{sQ}[k]-R_s i_{sQ}[k]-\sigma L_s\left[\frac{1-z^{-1}}{Ts}\right]i_{sQ}[k]\right] \qquad (3.79)$$

(3.79) can also be written as:

$$\left[\frac{\psi_{rd}[k]-\psi_{rd}[k-1]}{T_s}\right]=\frac{L_r}{L_m}\left[v_{sD}[k]-R_s i_{sD}[k]-\sigma L_s\left[\frac{i_{sD}[k]-i_{sD}[k-1]}{T_s}\right]\right]$$

$$\left[\frac{\psi_{rq}[k]-\psi_{rq}[k-1]}{T_s}\right]=\frac{L_r}{L_m}\left[v_{sQ}[k]-R_s i_{sQ}[k]-\sigma L_s\left[\frac{i_{sQ}[k]-i_{sQ}[k-1]}{T_s}\right]\right] \qquad (3.80)$$

Rearranging (3.80) gives:

$$\psi_{rd}[k]=\psi_{rd}[k-1]+\frac{L_r}{L_m}\left[(v_{sD}[k]-R_s i_{sD}[k])T_s-\sigma L_s(i_{sD}[k]-i_{sD}[k-1])\right]$$

$$\psi_{rq}[k]=\psi_{rq}[k-1]+\frac{L_r}{L_m}\left[(v_{sQ}[k]-R_s i_{sQ}[k])T_s-\sigma L_s(i_{sQ}[k]-i_{sQ}[k-1])\right] \qquad (3.81)$$

Equation (3.81) represents the discrete-time reference model of the rotor flux linkage MRAS observer.

3.4.3.2 Discrete-time adaptive model

The continues-time equations that represent the adaptive model in the rotor reference frame are:

$$p\psi_{rd}^{r}=-\frac{1}{T_r}\psi_{rd}^{r}+\frac{L_m}{T_r}i_{sD}^{r}$$

$$p\psi_{rq}^{r}=-\frac{1}{T_r}\psi_{rq}^{r}+\frac{L_m}{T_r}i_{sQ}^{r} \qquad (3.82)$$

Applying Laplace transformation on (3.82) yields:

$$s\psi_{rd}^{r} = -\frac{1}{T_r}\psi_{rd}^{r}(s) + \frac{L_m}{T_r}i_{sD}^{r}(s)$$

$$s\psi_{rq}^{r} = -\frac{1}{T_r}\psi_{rq}^{r}(s) + \frac{L_m}{T_r}i_{sQ}^{r}(s) \tag{3.83}$$

According to Euler backward transformation: $s = \frac{1-z^{-1}}{Ts}$, and (3.72) becomes:

$$\left[\frac{1-z^{-1}}{Ts}\right]\psi_{rd}^{r}[k] = -\frac{1}{T_r}\psi_{rd}^{r}[k] + \frac{L_m}{T_r}i_{sD}^{r}[k]$$

$$\left[\frac{1-z^{-1}}{Ts}\right]\psi_{rq}^{r}[k] = -\frac{1}{T_r}\psi_{rq}^{r}[k] + \frac{L_m}{T_r}i_{sQ}^{r}[k] \tag{3.84}$$

(3.84) can also be written as:

$$\left[\frac{\psi_{rd}[k]-\psi_{rd}[k-1]}{T_s}\right] = -\frac{1}{T_r}\psi_{rd}^{r}[k] + \frac{L_m}{T_r}i_{sD}^{r}[k]$$

$$\left[\frac{\psi_{rq}[k]-\psi_{rq}[k-1]}{T_s}\right] = -\frac{1}{T_r}\psi_{rq}^{r}[k] + \frac{L_m}{T_r}i_{sQ}^{r}[k] \tag{3.85}$$

Rearranging (3.85) gives:

$$\psi_{rd}[k] = (1-\frac{T_s}{T_r})\psi_{rd}^{r}[k] + \frac{T_sL_m}{T_r}i_{sD}^{r}[k]$$

$$\psi_{rd}[k] = (1-\frac{T_s}{T_r})\psi_{rq}^{r}[k] + \frac{T_sL_m}{T_r}i_{sQ}^{r}[k] \tag{3.86}$$

Equation (3.86) represents the discrete-time adaptive model of the rotor flux linkage MRAS observer.

3.4.3.3 Discrete-time adaption mechanism

As it has been discussed previously, the speed tuning signal ε_ω can be minimized by applying a *PI* controller, and, therefore, it can be considered as the adaption mechanism of the observer.

The continues-time *PI* is written as:

$$\hat{\omega}_r(t) = K_p\varepsilon_\omega(t) + K_i\int\varepsilon_\omega(t)dt = y_1(t) + y_2(t) \tag{3.87}$$

where $y_1(t) = K_p\varepsilon_\omega(t)$ and $y_2(t) = K_i\int\varepsilon_\omega(t)dt$.

Applying Laplace transformation gives:

$$\hat{\omega}_r(s) = K_p\varepsilon_\omega(s) + \frac{K_i}{s}\varepsilon_\omega(s) = Y_1(s) + Y_2(s) \tag{3.88}$$

where $Y_1(s), Y_2(s)$ are the Laplace transformation of $y_1(t)$ and $y_2(t)$, respectively.

From Euler backward transformation and by substituting *s* by $\frac{1-z^{-1}}{Ts}$ the following can be written:

$$y_1[k] = K_p\varepsilon_\omega[k]$$

$$y_2[k] = K_iT_s\varepsilon_\omega[k] + y_2[k-1]$$

$$\hat{\omega}_r[k] = y_1[k] + y_1[k] \tag{3.89}$$

3.5 Summary

This chapter has presented the modelling of the IMs using space vector principle and the principle of vector control has also been explained. It has also discussed the flux linkage MRAS observer modelling in both continues and discrete time domains.

CHAPTER 4

Model Predictive MRAS Speed Observers

4.1 Introduction

As has been discussed in the previous chapter, in the classical MRAS based speed observers, a fixed-gain PI controller is employed in the adaptation mechanism to produce the estimated position or speed. This is because of its simple structure and ability to generate a satisfactory performance over a wide range of speeds. However, at low speeds, inverter nonlinearities and machine parameter variation become more dominant. As a result, the fixed gain PI may not be able to maintain the system stability or at least to provide the required performance. Moreover, the tuning of these PI gains is not an easy task and little effort has been devoted in the literature to address this problem.

In the first part of this chapter, a novel MP-MRAS speed observer for sensorless vector control IM drives is introduced to solve the problems associated with the adaption mechanism design. The finite control set model predictive control concept is incorporated in the design of the new observer. In the proposed scheme, the adaptation mechanism is based on solving an optimization problem with the objective of minimizing the speed tuning error signal of the MRAS observer over a finite number of rotor position angles. A rotor position search algorithm is developed to ensure that the optimal position is obtained at each sampling time. The computational complexity of the proposed scheme is evaluated and a modified method is employed to reduce its execution time to make it suitable for practical implementation. The performance of the proposed predictive observer is tested both experimentally and by simulation using Matlab R2013a and a 2.2kW IM drive which employs FOC as the motor control strategy. A detailed comparison between the proposed scheme and the classical rotor flux MRAS observer is clarified in the next chapters. Results show the superior performance of the proposed scheme at different low speed operating conditions including regeneration and improved robustness against motor parameter variations.

In the second part of the chapter, a voltage compensating method is introduced to the reference model of the MP-MRAS observer to improve the system robustness. The proposed compensating method runs in parallel with the speed estimation algorithm, and it takes advantage of comparing the fluxes at the output of the adaptive and reference models which are calculated based on the current and voltage models of the IM respectively. This compensates for any disturbance taking place in the reference model, and hence, results in reliable speed estimation. In this part also, a new method to extract motor speed from the rotor position signal is introduced. This method employs an adaptive low pass filter (LPF) which can adapt its time constant depending on the dynamic state of the system to allow higher filtering quality at steady state and faster dynamic response during the transients. The simulation and experimental results in the next chapters confirm the effectiveness of the proposed scheme for sensorless speed control of IM drives.

This chapter is divided into three parts; the first part presents the principle of model predictive controllers (MPC) and finite control set model predictive controllers (FCS-MPC). The second part introduces the principle of the proposed MP-MRAS observers. The final part discusses the proposed voltage compensating method and speed extraction mechanism.

4.2 Model Predictive Controllers (MPC)

Model Predictive Controllers (MPC) use the system model to predict the future behaviour of one or more of the system variables over a prediction horizon, which is an integer multiple of the sampling period. A cost function is then used to evaluate these predictions and to choose the one that minimize the cost function. This leads to a sequence of future control actions over the prediction horizon, where only the first value of this sequence is applied. This algorithm is repeated at every sampling period during the operation [86].

The main advantage of using MPC is that the system constraints and nonlinearities can be easily contained in the cost function and the controller design. Unfortunately, MPCs require a large number of calculations to solve the optimization problem which make them unsuitable for power converter control and drive applications. This is because in such applications, the control problem is required to be solved in real time within one sampling time, which is usually too short to process the MPC algorithm [86].

Many solutions have been suggested to overcome the computational problem associated with MPCs, which has led to different types of MPCs. One solution is to solve the optimization problem off-line and then replacing it by a search tree in the real time application [87]. The Generalized Predictive Control (GPC) is another solution, where the optimization problem is solved analytically to obtain a linear control [88]. The most feasible method that has been widely applied in power converter and drive applications is the FCS-MPC [81, 89-93]. These controllers use the discrete nature of the power converter (Fig. 4.1) in solving the optimization problem, which means that the system behaviour is only predicted at the switching states of the power converter and the cost function is evaluated at these predictions.

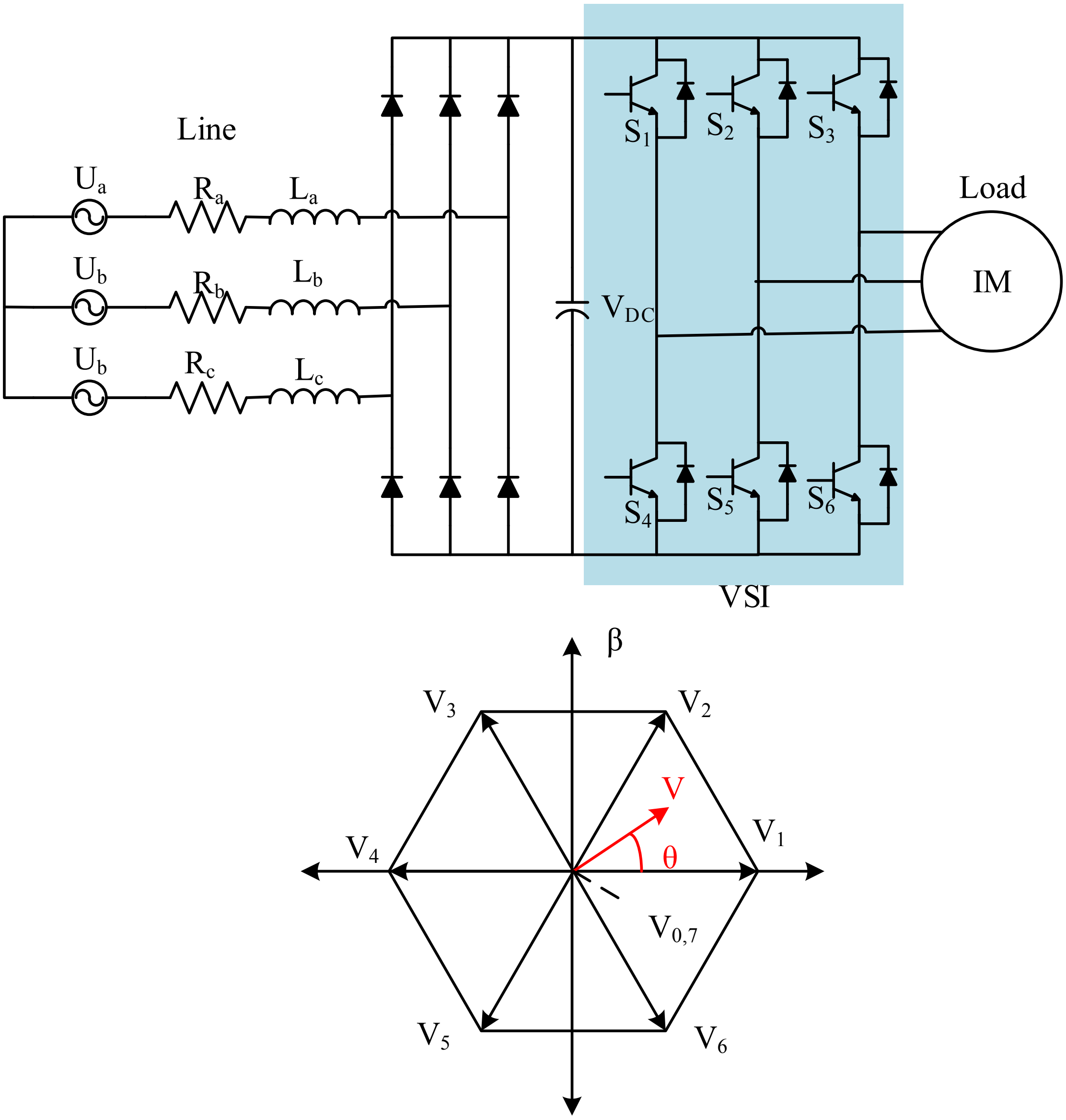


Fig. 4.1 Block diagram of generic three phase VSI inverter

4.2.1 Finite control set model predictive controller

Generally speaking, the control problem of any power converter or drive system can be simply defined as choosing the proper control action *S(t)* (which is usually the gate signals of the power switches), that can drive a system variable *x(t),* as close as possible, to a demanded value $x^*(t)$ of the same variable [86].

If a system with a finite number of control actions *n* is considered, the measured value of the variable *x(k)* at the time instant *k* can be evaluated along with the control action S_n to predict all possible transients of the variable $x_i^p(k+1)$ at the next sampling period, according to a prediction function *f* which is derived from the discrete model of the controlled system. This means the predictions $x_i^p(k+1)$ can be calculated as:

$$x_i^p(k+1) = f\{x(k), S_i\} \quad \text{for } i = 1, 2, \ldots, n \tag{4.1}$$

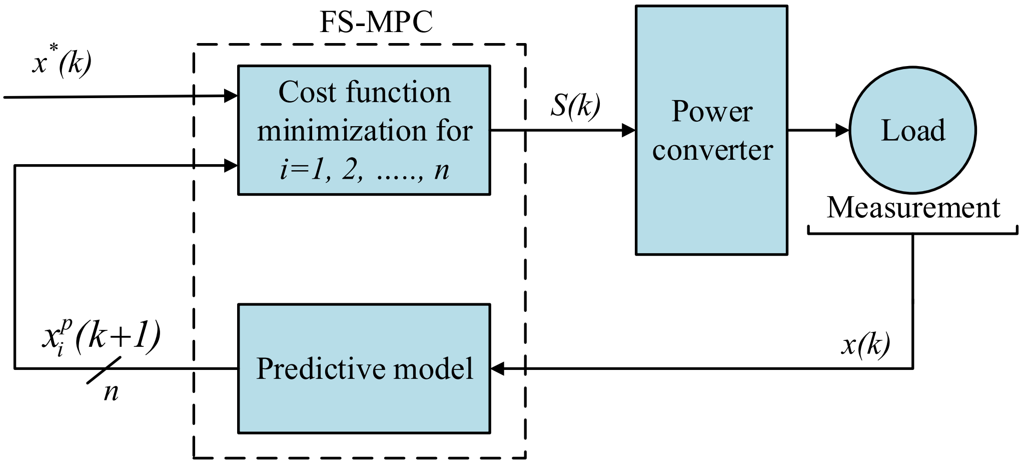


Fig. 4.2 Block diagram of FCS-MPC

To choose a proper control action, a cost function f_g is defined, which depends on both the prediction $x_i^p(k+1)$ and the future reference value $x_i^*(k+1)$ to calculate the different costs g_i corresponding to each of the predictions:

$$g_i = f_g\{x_i^p(k+1), x_i^*(k+1)\} \quad \text{for } i = 1, 2, \ldots, n \tag{4.2}$$

It can be noticed from (4.2) that, in order to find the different costs, the desired value of the variable $x_i^*(k+1)$ at the next sample *k+1* is required. For systems which have a slow dynamic response relative to the system sampling period T_s, the desired value x^* can be considered constant over one sampling period T_s which means $x_i^*(k+1)=x_i^*(k)$, whereas for highly dynamic systems, $x_i^*(k+1)$ should be estimated via an appropriate method. The cost function in its simplest form can be considered equal to the absolute value of the error between the predicted and the demanded values $g_i = | x_i^*(k+1) - x_i^p(k+1) |$. As a result, by evaluating the cost function at the *n* different predictions, *n* different costs are obtained and the control action associated with the minimum cost, i.e. $min(g_i, i=1,2,...,n)$, is chosen as the output of the controller to control the system variable *x*. Fig. 4.2 presents a block diagram of the FCS-MPC system.

As a result, in contrast to the conventional model predictive controllers, FCS considers the discrete nature of the inverter in solving the control optimization problem [94]. This method therefore has the advantages of both simplicity and design flexibility making it attractive to electric drives applications [79].

4.3 The Proposed Model Predictive MRAS Observer

As mentioned previously, the operational principle of the FCS-MPCs is used to design the adaptation mechanism in the proposed MP-MRAS speed observers. An optimization problem is formulated to find the rotor position in order to minimize a cost function, which is the speed tuning signal (3.75) in the case of the MRAS observer.

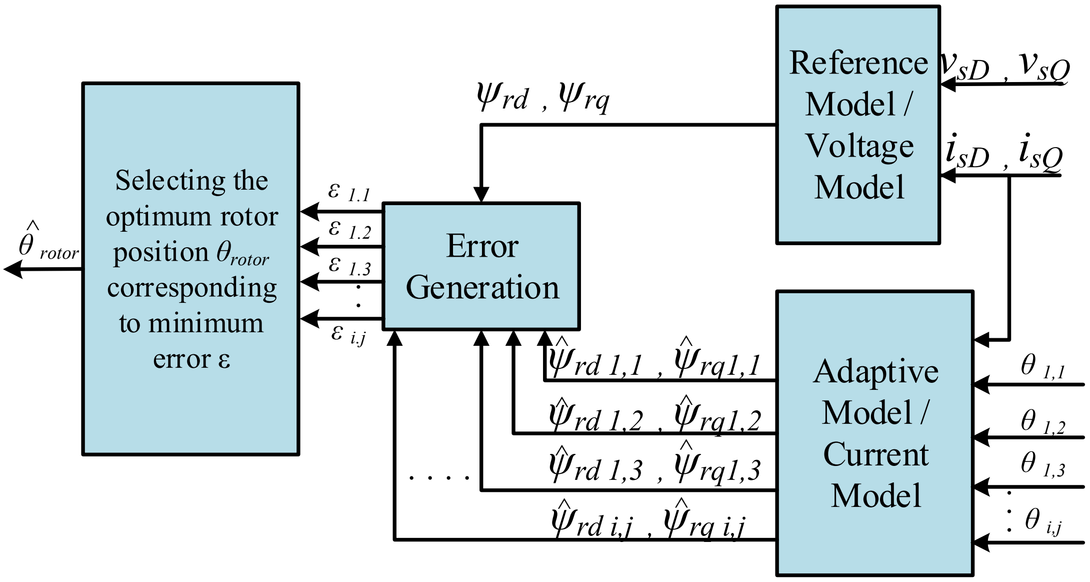

Reference Model / Voltage Model
v_{sD} , v_{sQ}
i_{sD} , i_{sQ}
ψ_{rd} , ψ_{rq}
Error Generation
Selecting the optimum rotor position θ_{rotor} corresponding to minimum error ε
$\hat{\theta}_{rotor}$
$\varepsilon_{1.1}$
$\varepsilon_{1.2}$
$\varepsilon_{1.3}$
$\varepsilon_{i.j}$
$\hat{\psi}_{rd\,1,1}$, $\hat{\psi}_{rq1,1}$
$\hat{\psi}_{rd\,1,2}$, $\hat{\psi}_{rq1,2}$
$\hat{\psi}_{rd\,1,3}$, $\hat{\psi}_{rq1,3}$
$\hat{\psi}_{rd\,i,j}$, $\hat{\psi}_{rq\,i,j}$
Adaptive Model / Current Model
$\theta_{1,1}$
$\theta_{1,2}$
$\theta_{1,3}$
$\theta_{i,j}$

Fig. 4.3 Block diagram of the proposed MP-MRAS observer

In contrast to the FCS-MPC, the rotor position, which varies continuously between 0 and 360°, does not have the same discrete nature as the inverter output. Therefore a search method is applied to discretize the rotor position into a finite number of positions to allow evaluating the cost function at each of these discrete positions. This search is performed within an iteration based process. The block diagram of the proposed MP-MRAS observer is shown in Fig. 4.3. The flow chart of the proposed search algorithm is shown in Fig. 4.4. The algorithm starts by calculating the reference model outputs ψ_{rD}, ψ_{rQ} from the stator voltages and currents. The discretization of the rotor position begins by starting from an initial base angle $\theta_{base,0}$ and then displacing this angle by a displacement ($\Delta\theta_i$) which is calculated as follow:

$$\Delta\theta_i = 45^o \cdot 2^{-i} \tag{4.3}$$

where i is the order of the current iteration.

The displacement of the base angle θ_{base} within each iteration is carried out to get eight discrete rotor positions as follow:

$$\theta_{i,j} = \theta_{base} + \Delta\theta_i . (j - 4) \tag{4.4}$$

where j is the order of the displacement.

In the initial iteration (i=0), the base angle θ_{base} is chosen to be 0° with $\Delta\theta=45^o$ according to (4.3). Applying (4.4) will produce eight discrete positions: 0°, 45°, 90°, 135°, 180°, -45°, -90°, -135°. Each of these discrete positions ($\theta_{i,j}$) is used to calculate the adaptive model outputs corresponding to each individual position ($\hat{\psi}_{\mathrm{rd}_{i,j}}$ and $\hat{\psi}_{\mathrm{rq}_{i,j}}$). Consequently the cost function, $\varepsilon_{i,j}$ in (3.75), is calculated for each position as follows:

$$\varepsilon_{i,j} = \hat{\psi}_{rd_{i,j}} \psi_{rq} - \hat{\psi}_{rq_{i,j}} \psi_{rd} \tag{4.5}$$

This leads to eight different cost functions corresponding to each of these angles. The angle corresponding to the minimum cost function of the eight positions is chosen as the base or starting point $\theta_{base,\,1}$ for the next iteration.

At the next iteration ($i=1$), the angle displacement is decreased to $\Delta\theta_1 = 45^o \times 2^{-1} = 22.5^o$, which increases the search accuracy by a factor of 2. The search then starts again from the new base angle $\theta_{base,1}$ to find the angle that generates the minimum cost function in the second iteration. Fig. 4.5 shows the initial and first steps of the search algorithm.

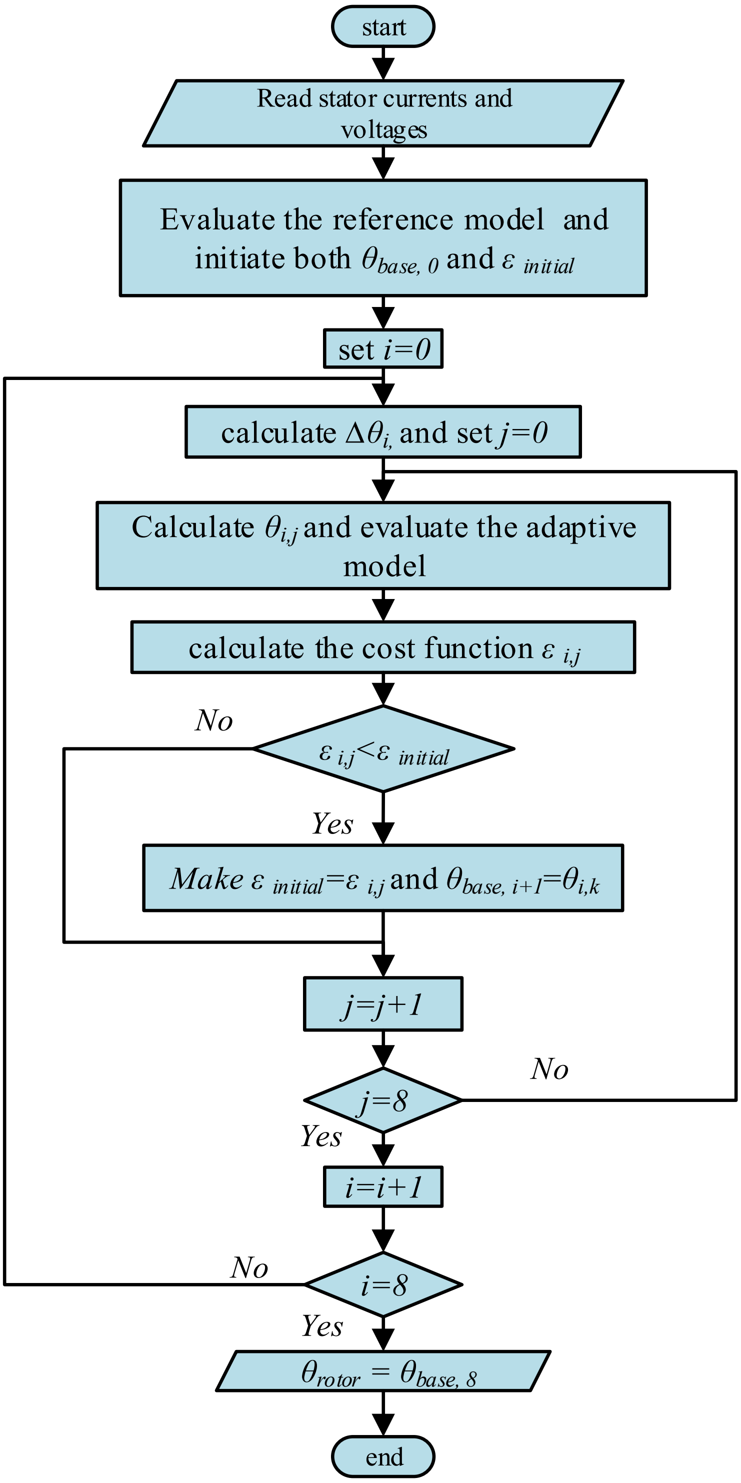

start
Read stator currents and voltages
Evaluate the reference model and initiate both $\theta_{base,\,0}$ and $\varepsilon_{initial}$
set $i=0$
calculate $\Delta\theta_{i,}$ and set $j=0$
Calculate $\theta_{i,j}$ and evaluate the adaptive model
calculate the cost function $\varepsilon_{i,j}$
$\varepsilon_{i,j}<\varepsilon_{initial}$
No
Yes
Make $\varepsilon_{initial}=\varepsilon_{i,j}$ and $\theta_{base,\,i+1}=\theta_{i,k}$
$j=j+1$
$j=8$
No
Yes
$i=i+1$
$i=8$
No
Yes
$\theta_{rotor}=\theta_{base,\,8}$
end

Fig. 4.4 Flowchart of the proposed rotor position search algorithm

After each iteration, the search algorithm gets closer to the optimal solution, and by the end of the 8th iteration (i=7 and $\Delta\theta_7=0.35^o$), the optimal rotor position can be found with 0.35°

accuracy. Therefore, by running this algorithm, it can be assured that the optimal rotor position, which produces the minimum cost function throughout the search space, is selected as the output of the observer.

As described previously, the output of the proposed scheme is the rotor position, and to extract the speed signal the following procedure is applied:

The change in rotor position over the last sampling period is calculated from:

$$\Delta\theta = \theta_{rotor}(k) - \theta_{rotor}(k-1) \tag{4.6}$$

where k is the current time sample.

This change is recorded over 200 samples and the average value is obtained by applying:

$$\Delta\theta_{ave} = \frac{1}{200}\sum_{n=1}^{200}\Delta\theta_n \tag{4.7}$$

The speed is finally found by dividing the average by the sampling period. The conversion to rad/sec is considered here also.

$$N = \frac{2\pi}{60}\cdot\frac{\Delta\theta_{average}}{T_s} \tag{4.8}$$

where N is the rotor speed in rpm.

A drawback of the proposed method is the high computational effort required to run the search algorithm eight times in each sampling period. However, the rotor position, as a mechanical variable, changes relatively slowly and hence it does not vary significantly between two time samples. Therefore, instead of initiating the search algorithm in each sampling period with zero angle ($\theta_{base,0}=0)$, it can be initialized by the output of the algorithm in the last sampling instant $\theta_{base,0}=\theta_{rotor}(k-1)$. As a result, the number of the iterations required by the search algorithm to find the optimal solution can be significantly reduced as the search is performed only around the previous rotor position. This simplified scheme is referred to as “modified-predictive”.

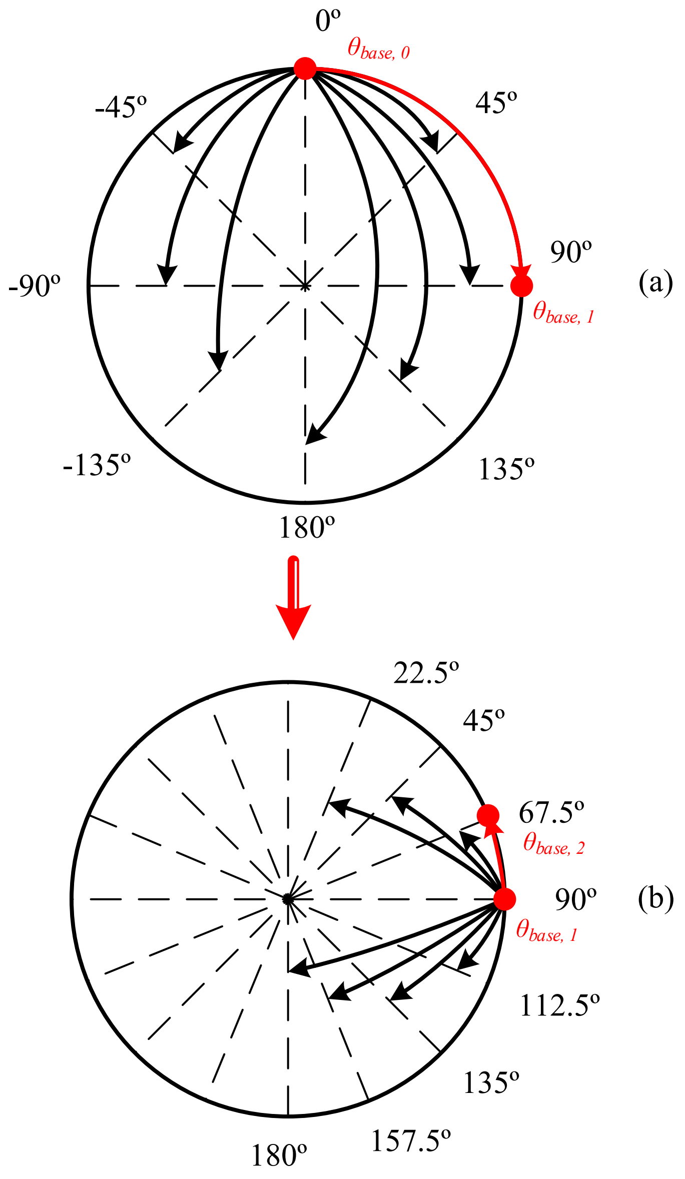


Fig. 4.5 Schematic representation of the first two steps of the proposed search algorithm. (a) Initial iteration. (b) First iteration.

Experimentally, it was found that only the last iteration loop (i=7) is required to find the rotor position using the modified-predictive scheme without affecting the estimation accuracy. This significantly reduces the execution time of the proposed scheme from *103 μs* to *39 μs*. For comparison purpose, Table 4.1 shows the execution times for the two versions of the proposed predictive scheme in addition to the *PI*-based classical MRAS observer. It should be mentioned here that these times are specific for the TMS320F28335 floating point microcontroller (150 MHz clock) used in the experiments and it can be further reduced if a faster microcontroller is applied.

From now on the term “MP-MRAS observer” will be used to refer to the modified scheme with the reduced execution time.

The proposed predictive scheme applies an iterative search method to find the rotor position. This is fundamentally different from other MRAS observers available in the literature, such as those using PI, sliding mode and fuzzy logic adaptation mechanisms. The proposed method does not require any gain tuning like the aforementioned schemes which make the design of the observer much simpler and ensure the optimum operation of the observer at all operating speeds. Application of the proposed scheme always ensures that the speed tuning signal is driven to almost zero in each sampling period. The scheme is capable of achieving minimum error in one sampling time following any disturbance. This results in the proposed scheme having a significant advantage over other approaches.

Symbol	Execution Time
Control Sampling Time	100 μs
PI	14 μs
MP-MRAS	103 μs
Modified MP-MRAS	39 μs

Table 4.1 The execution time of the different applied observers

4.4 Discussion of the search-based optimization algorithm

As it has been discussed, the proposed observer applies a search-based algorithm to find the rotor position corresponding to the minimum estimation error. In this section, the relationship between the estimated flux angular position and the speed tuning signal (the estimation error) will be investigated in more details. This investigation will be deployed next to explain the methodology applied to design the search algorithm applied in the thesis.

The speed tuning signal (the estimation error) is given by:

$$\varepsilon_{\omega} = \psi_{rq}\hat{\psi}_{rd} - \psi_{rd}\hat{\psi}_{rq} \tag{4.9}$$

The actual and estimated rotor flux vector coordinates can be expressed as:

$$\psi_{rd} = \psi\cos(\theta_e)$$
$$\psi_{rq} = \psi\sin(\theta_e) \tag{4.10}$$

$$\hat{\psi}_{rd} = \hat{\psi}\cos(\hat{\theta}_e)$$
$$\hat{\psi}_{rq} = \hat{\psi}\sin(\hat{\theta}_e) \tag{4.11}$$

Substituting (4.10) and (4.11) into (4.9) yields:

$$\varepsilon_{\omega} = \psi\sin(\theta_e)\cdot\hat{\psi}\cos(\hat{\theta}_e) - \psi\cos(\theta_e)\cdot\hat{\psi}\sin(\hat{\theta}_e)$$
$$\varepsilon_{\omega} = \psi\hat{\psi}\sin(\theta_e - \hat{\theta}_e) \tag{4.12}$$

From (4.12) it can be noticed that the relationship is sinusoidal between the speed tuning signal ε_{ω} and the estimated position $\hat{\theta}_e$ for a given rotor flux angle θ_e. In other words, the speed tuning signal will be changing sinusoidaly over the search process as the position iteration occurs. This is shown in Fig. 4.6.

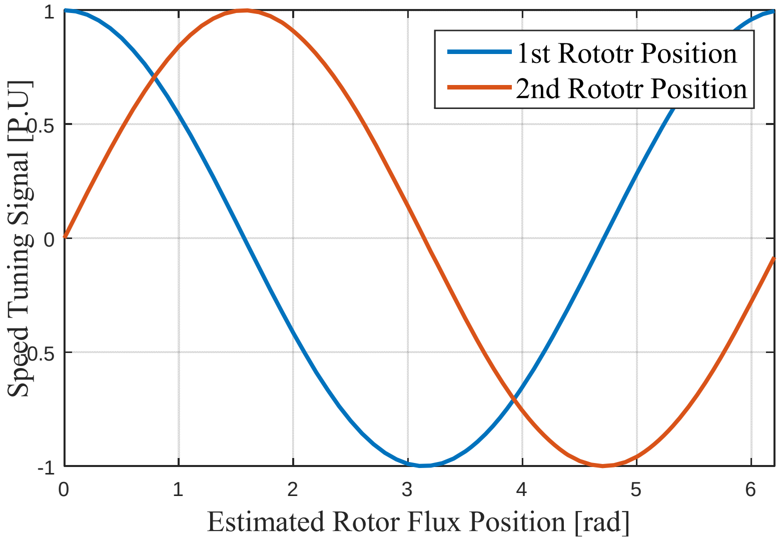


Fig. 4.6 Speed tuning signal ε versus the estimated rotor position $\hat{\theta}_e$

1^{st} rotor position is $\theta_e = \frac{\pi}{2}$, 2^{nd} rotor position is $\theta_e = \pi$

After the estimation error is found as a function of the estimated rotor flux position, the design of the proposed search algorithm can be discussed.

As it has been found, the estimation error ε_{ω} has a sinusoidal relationship with estimated rotor flux position $\hat{\theta}_e$. This implies two different points. The first one is that ε_{ω} does not have a local minima, and, therefore, it is not required to apply a complicated estimation method to find the point at which this error is equal to zero. The second point is that the sinusoidal relationship means that there are two different points at which the estimation error ε_{ω} becomes zero. For example from Fig. 4.6, in the case when $\theta_e = \pi/2$, it can be seen that ε_{ω} becomes zero at $\hat{\theta}_e = \pi/2$ and $\hat{\theta}_e = 3\pi/2$. This makes it difficult to tell which of the two values represent the solution. To solve this problem, instead of searching the point at which the estimation error becomes zero, the position $\hat{\theta}_{e(\max)}$ at which the error becomes at the maximum can be searched instead. The real solution will be then shifted by $\pi/2$ from this point. This means:

$$\hat{\theta}_e = \hat{\theta}_{e(\max)} + \frac{\pi}{2} \tag{4.13}$$

The same procedure was followed in this thesis in order to find the estimated rotor position corresponding to the minimum speed tuning signal.

4.5 Improved Model Predictive MRAS Observer

As will be discussed in the next chapters, the MP-MRAS observer gives a better performance at different operating conditions, including regeneration, in comparison with classical MRAS-based observers. However, the experimental results shows oscillations in the estimated rotor speed, especially at light loading conditions. Furthermore, due to the use of the voltage equation in the reference model, the scheme remains sensitive to the variations in the machine parameters.

In this section, a new speed extraction method is introduced to the design of the MP-MRAS observer to reduce the oscillations in the estimated speed at light loading conditions. Furthermore, a voltage compensating method is employed in the reference model of the observer to address the problem of sensitivity to motor parameters variation.

4.5.1 Speed extraction using an adaptive low pass filter

As mentioned previously, the MP-MRAS observer presented shows oscillations in the estimated rotor speed especially at light loading conditions. In this section, a new method to extract motor speed from the rotor position signal is introduced. This method employs an adaptive low pass filter (LPF) which can adapt its time constant depending on the dynamic state of the system to allow higher filtering quality at steady state and faster dynamic response during transient operation.

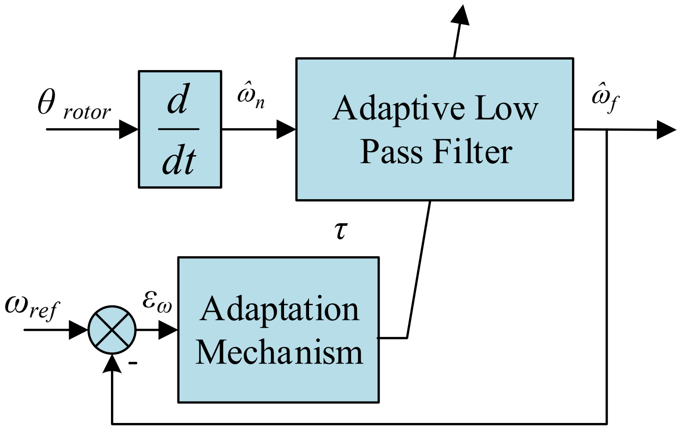


Fig. 4.7 Block diagram of the proposed speed extraction method

The operation of the proposed method starts by differentiating the rotor position signal θ_r, which results from the predictive observer algorithm (Fig. 4.7). The outcome of this differentiation is the rotor angular velocity ω_n, which contains high level oscillations as a result of the differentiation. These oscillations are reduced by applying an adaptive LPF. The filter adaptation mechanism is shown in Fig. 4.8.

The filter adaptation mechanism reads the error between the reference and the estimated speeds ε_ω and depending on its value it determines the system dynamic state. This error is close to zero during the steady state and it increases when a disturbance in the system takes place. When the error ε_ω exceeds a set value, the filter adaptation mechanism decreases the time constant instantaneously to τ_{low} to allow a faster dynamic response. As a result, the motor speed moves toward the reference speed faster and the error ε_ω starts to decrease. Once the error ε_ω becomes lower than the same previous set value, the adaption mechanism starts to gradually reincrease the filter time constant until it reaches τ_{high}.

It has been found experimentally that if the filter time constant is increased suddenly, the system can lose its stability because of the spikes that appear at the output of the filter.

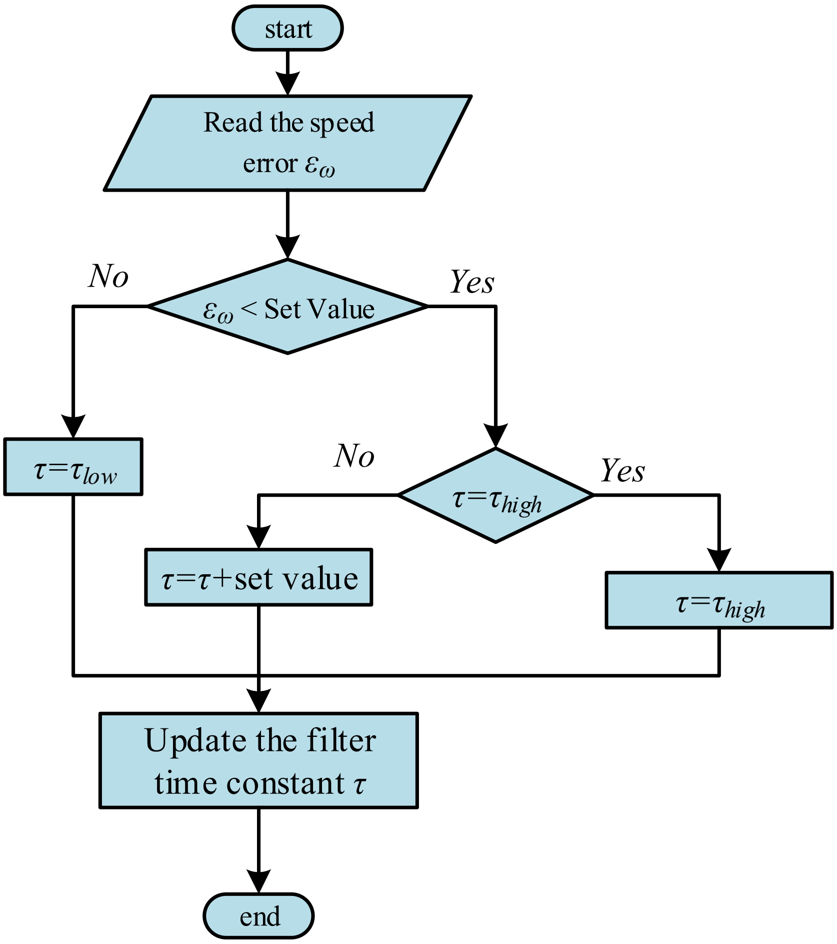

start
Read the speed error ε_ω
No
Yes
ε_ω < Set Value
$\tau=\tau_{low}$
No
Yes
$\tau=\tau_{high}$
$\tau=\tau$+set value
$\tau=\tau_{high}$
Update the filter time constant τ
end

Fig. 4.8 The flow chart of the LPF adaptation mechanism

4.5.2 The reference model voltage compensating method

As mentioned in the beginning of this chapter, due to the use of the voltage equation in the reference model, the MP-MRAS observer shows sensitivity to the motor parameter variations. In this section a voltage compensating method is introduced to the reference model to improve the system robustness.

The proposed compensating method runs in parallel with the speed estimation algorithm, and it takes advantage of comparing the fluxes at output of the adaptive and reference models which are calculated based on the current and voltage models of the IM respectively. This can compensate for any disturbance taking place in the reference model, and hence, results in reliable speed estimation. The block diagram of the proposed method is shown in Fig. 4.9.

The fluxes in the adaptive model are calculated in the stationary reference frame based on the current model as:

$$\frac{d}{dt}\hat{\psi}_{rd} = \frac{L_m}{T_r} i_{sD} - \frac{1}{T_r}\hat{\psi}_{rd} - \hat{\omega}_r \hat{\psi}_{rq}$$

$$\frac{d}{dt}\hat{\psi}_{rq} = \frac{L_m}{T_r} i_{sQ} - \frac{1}{T_r}\hat{\psi}_{rq} + \hat{\omega}_r \hat{\psi}_{rd} \tag{4.14}$$

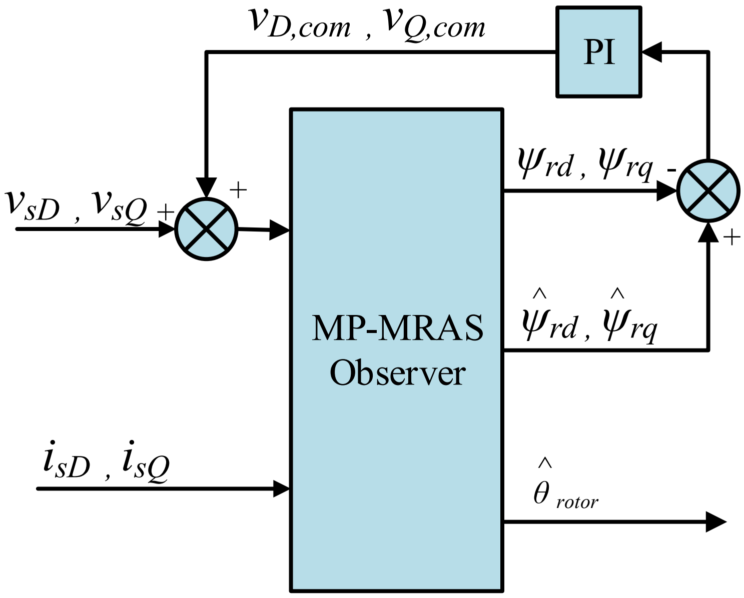

$v_{D,com}$, $v_{Q,com}$
PI
v_{sD} , v_{sQ}
+
+
ψ_{rd} , ψ_{rq}
-
+
MP-MRAS Observer
$\hat{\psi}_{rd}$, $\hat{\psi}_{rq}$
i_{sD} , i_{sQ}
$\hat{\theta}_{rotor}$

Fig. 4.9 Block diagram of the compensated system

Note that the stator resistance R_s does not appear in the equations, and L_m is divided by T_r which makes the effect of these parameters negligible in the adaptive model.

The reference model calculates the same flux components based on the voltage model in the stationary reference frame as:

$$\frac{d}{dt}\psi_{rd} = \frac{L_r}{L_m}(v_{sD} - R_s i_{sD} - \sigma L_s \frac{d}{dt} i_{sD})$$

$$\frac{d}{dt}\psi_{rq} = \frac{L_r}{L_m}(v_{sQ} - R_s i_{sQ} - \sigma L_s \frac{d}{dt} i_{sQ}) \qquad (4.15)$$

From (4.14) and (4.15) it can be noticed that both R_s, and L_m variation has greater effect on the flux calculation in the reference model compared to the adaptive model. To reduce this effect, two *PI* controllers are used to minimize the difference between the two models outputs, and, hence, to compensate for the parameter variation. This results in two voltage compensating terms as:

$$\mathrm{v}_{D,com} = \left(\frac{K_i}{s} + K_p\right) \cdot \varepsilon_D$$

$$\mathrm{v}_{Q,com} = \left(\frac{K_i}{s} + K_p\right) \cdot \varepsilon_Q \qquad (4.16)$$

where K_p and K_i are the gains of the *PI* controller in the voltage compensating method and $\varepsilon_{D,}$ ε_Q are the errors between the reference and the adaptive models fluxes which are calculated as:

$$\varepsilon_D = \hat{\psi}_{\mathrm{r}d} - \psi_{\mathrm{r}d}$$

$$\varepsilon_Q = \hat{\psi}_{\mathrm{r}q} - \psi_{\mathrm{r}q} \qquad (4.17)$$

These compensating terms are added finally to the voltages in the input of the reference model which becomes:

$$\frac{d}{dt}\psi_{rd} = \frac{L_r}{L_m}(v_{sD} + \mathrm{v}_{D,com} - R_s i_{sD} - \sigma L_s \frac{d}{dt} i_{sD})$$

$$\frac{d}{dt}\psi_{rq} = \frac{L_r}{L_m}(v_{sQ} + \mathrm{v}_{Q,com} - R_s i_{sQ} - \sigma L_s \frac{d}{dt} i_{sQ}) \qquad (4.18)$$

To mathematically prove the stability of the proposed method, two disturbance terms D_D and D_Q are added to the reference model equations to represent the motor parameter variations. As a result, the reference model equations become:

$$\psi_{\mathrm{rd}} = \frac{L_r}{L_m}\int (v_{sD} + v_{D,com} - R_s i_{sD} - \sigma L_s \frac{di_{sD}}{dt} + D_D)$$

$$\psi_{\mathrm{r}q} = \frac{L_r}{L_m}\int (v_{sQ} + v_{Q,com} - R_s i_{sQ} - \sigma L_s \frac{di_{sQ}}{dt} + D_Q) \tag{4.19}$$

Substituting (4.19) in (4.17) yields:

$$\varepsilon_D = \hat{\psi}_{\mathrm{rd}} - \frac{L_r}{L_m}\int (v_{sD} + v_{D,com} - R_s i_{sD} - \sigma L_s \frac{di_{sD}}{dt} + D_D)$$

$$\varepsilon_Q = \hat{\psi}_{\mathrm{rq}} - \frac{L_r}{L_m}\int (v_{sQ} + v_{Q,com} - R_s i_{sQ} - \sigma L_s \frac{di_{sQ}}{dt} + D_Q) \tag{4.20}$$

Since this compensator operates in parallel with the speed observer and because the MP-MRAS observer can drive the speed tuning error to zero within one PWM cycle (as discussed in the previous section), the following assumption can be made:

$$\hat{\psi}_{\mathrm{r}d} = \frac{L_r}{L_m}\int (v_{sD} - R_s i_{sD} - \sigma L_s \frac{di_{sD}}{dt})$$

$$\hat{\psi}_{\mathrm{r}q} = \frac{L_r}{L_m}\int (v_{sQ} - R_s i_{sQ} - \sigma L_s \frac{di_{sQ}}{dt}) \tag{4.21}$$

By substituting (4.21) in (4.20) and removing the similar terms the following can be obtained:

$$\varepsilon_D = \frac{L_r}{L_m}\int (-v_{D,com} - D_D)$$

$$\varepsilon_Q = \frac{L_r}{L_m}\int (-v_{Q,com} - D_Q) \tag{4.22}$$

Substituting (4.16) in (4.22) results in:

$$\varepsilon_D = \frac{L_r}{L_m}\int \{-k_p \varepsilon_D - \int (k_i \varepsilon_D) dt - D_D\} dt$$

$$\varepsilon_Q = \frac{L_r}{L_m}\int\{-k_p\varepsilon_Q - \int(k_i\varepsilon_Q)dt - D_Q\}dt \tag{4.23}$$

To get rid of the integration in (4.23) the second derivation is taken:

$$\ddot{\varepsilon}_D = -\frac{L_r}{L_m}k_p\dot{\varepsilon}_D - \frac{L_r}{L_m}k_i\varepsilon_D - \frac{L_r}{L_m}\dot{D}_\alpha$$

$$\ddot{\varepsilon}_Q = -\frac{L_r}{L_m}k_p\dot{\varepsilon}_Q - \frac{L_r}{L_m}k_i\varepsilon_Q - \frac{L_r}{L_m}\dot{D}_Q \tag{4.24}$$

The dynamic response of the distortion terms D_D and D_Q is far too slow compared to the drive system dynamic response since they represent the motor parameter variations. This results in $\dot{D}_D = \dot{D}_Q \approx 0$ and the error dynamic equations become:

$$\ddot{\varepsilon}_D = -\frac{L_r}{L_m}k_p\dot{\varepsilon}_D - \frac{L_r}{L_m}k_i\varepsilon_D$$

$$\ddot{\varepsilon}_Q = -\frac{L_r}{L_m}k_p\dot{\varepsilon}_Q - \frac{L_r}{L_m}k_i\varepsilon_Q \tag{4.25}$$

From (4.25), if the controller gain K_p is chosen to be positive, the error signals ε_D and ε_Q will converge to zero at steady state and the dynamic response of the error will be determined by the values of K_p and K_i.

From here on the observer with the voltage compensation and the new speed extraction method is referred to as the improved MP-MRAS observer and the original system without the compensation is referred to as the MP-MRAS observer.

4.6 Summary

In this chapter, a novel MP-MRAS rotor speed observer is proposed for sensorless IM drives. The new observer is based on the finite control set-model predictive control principle and applies an optimization approach to minimize the speed tuning error signal of the MRAS scheme. This eliminates the need for a PI controller in the adaptation mechanism. A search algorithm is employed to ensure that optimal rotor position is achieved in each sampling period that minimizes the error signal. A modification has been introduced to the proposed algorithm to reduce its computational complexity compared to conventional PI controller.

Due to the use of the voltage equation in the reference model, the MP-MRAS remains sensitive, to a certain extent, to the variations in the machine parameters. Therefore, a voltage compensating method is employed in the reference model of the MP-MRAS observer to solve problems associated with sensitivity to motor parameters variation. Furthermore, to reduce rotor speed oscillations at light loading conditions, an adaptive filter is employed in the speed extraction mechanism, where an adaptation mechanism is proposed to adapt the filter time constant depending on the dynamic state of the system.

CHAPTER 5

The Simulation Results

5.1 Introduction

To investigate the proposed speed estimation scheme, a model of a field oriented control-based drive system, was built with Matlab® R2013a Simulink® tools. Fig 5.1 shows the block diagram of the drive system model. The speed observer in the diagram can be the classical MRAS observer, the MP-MRAS observer or the improved MP-MRAS observer.

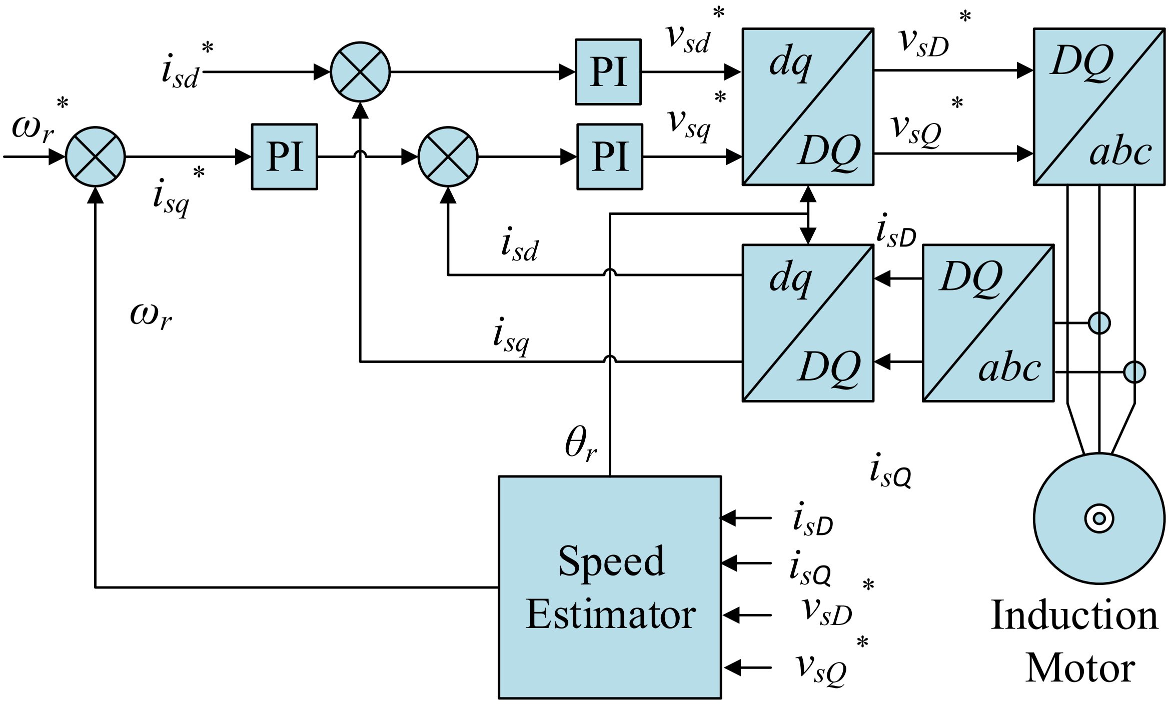


Fig. 5.1 The block diagram of the field oriented control drive system

In the simulation, it was assumed that the inverter is ideal which means that the IM is fed directly from the reference voltage. This was necessary to speed up the simulation which can significantly slow down by the PWM generating mechanism. However, this means that neither the inverter nonlinearity nor the dead time effects were taken into the consideration which can affect the estimation accuracy at low speeds.

Another approximation was taken in this simulation which is related to the voltage and current sensors. These sensors where considered ideal by neglecting the DC-offset, the noise and the wrong calibration effects on the measured signals, and hence on the the speed estimation. The effect of this approximation is less significant at high speeds where the signal to noise ratio is relatively high. But as the speed is reduced so do the voltage and current levels leading to a mismatch between the actual and simulated signals at very low speeds. This will be discussed further in chapter 7.

To create a more realistic simulation, the code was written by using C-language and S-function block. This ensures that the digital control code is exactly the same in both the simulation and the micro controller. In addition, the limitation of the microcontroller computational capability is considered. For example, the memory size allocated for each variable, the switching frequency, the way the sine and cosine function calculated, are all considered.

5.2 The Motor Simulation

The induction machine model in a stationary reference frame *(D,Q)* is performed. The 4-pole squirrel cage machine has the following parameters:

Symbol	Quantity	Value
R_s	Stator Resistance	2.35 Ω
R_r	Rotor Resistance	1.05 Ω
L_s	Stator Inductance	0.344209 H
L_r	Rotor Inductance	0.348197 H
L_m	Mutual Inductance	0.332090 H

Table 5.1 The induction motor electrical parameters

To obtain the mechanical rotor speed ω_r, the following mechanical model is applied, which determines the rotor speed, depending on the electromagnetic torque generated by the machine T_e, the load torque T_l, the machine moment of inertia J and the machine friction coefficient B as follows:

$$T_e - T_l = J\frac{d\omega_r}{dt} + B\omega_r \tag{5.1}$$

In (5.1) the machine moment of inertia J is obtained by applying the “run-down” test [95] and it was found to be equal to J=0.0139$Kg.m^{-2}$. The friction coefficient B was found the same way and it is equal 0.001128$Nm/rad.s^{-1}$.

5.3 The Control System Simulation

The field oriented control scheme is employed which is a very common control method for both induction and synchronous machines. As aforementioned, in this control algorithm the *d* axis of the synchronously rotating reference frame is chosen to be aligned with the rotor flux. The importance of this alignment is that it allows independent control of the magnetic flux and the electromagnetic torque produced in the machine. This is achieved by controlling *d-axis* and *q-axis* current components respectively in the synchronously rotating reference frame [2]. As in any other machine, the electromagnetic torque in the IM is equal to the cross product between the rotor flux and the stator current. This leads to the following: The rotor flux in the machine is produced by the *d* component of the stator current (i_{ds}) while the torque is produced by the *q* component of the stator current (i_{qs}) [2].

As it can be seen in Fig 5.1, the control system involves two interlocked control loops, a current loop and a speed loop. The purpose of the speed control loop is to force the measured speed (or the estimated speed in the sensorless systems) to follow the reference speed, which is achieved by feeding the speed error, the difference between the measured (estimated) and the reference speeds, into a PI controller. The speed PI controller will in-turn determine the value of the torque producing current component (i_{qs}^*) which will be able to correct the speed value. Similarly to the speed control loop, the flux and torque current components (i_{ds} and i_{qs}) are regulated by two other PI controllers. The outputs of these controllers are the stator reference voltages represented in a stationary reference frame (v_{ds}^* and v_{qs}^*).

Before the resulting three phase voltages can be applied to the machine terminals, two different transformations (Park^{-1} and Clarke^{-1}) are applied to find the three phase reference voltages. Likewise, after the three-phase currents are measured at the motor terminals, they are transformed via Clarke and Park transformations to (i_{sD},i_{sQ}) and (i_{sd},i_{sq}) to be compared with the reference currents (i_{sd}^*,i_{sq}^*).

As it can be seen in Fig 5.1, depending on the reference voltages (v_{sD}^*,v_{sQ}^*) and the currents (i_{sD},i_{sQ}), the speed observer (which can be either the classical or the model predictive MRAS observers) can calculate the rotor speed and the rotor flux position. These quantities are essential for both of the field orientation and reference frame transformations.

5.4 The Simulation Results

5.4.1 Performance comparison of the classical MRAS and the MP-MRAS observers

This part of the results evaluates the performance of the MP- MRAS observer in comparison with the classical rotor flux-based MRAS observer. Many tests are carried out in both open-loop and sensorless operation modes. The simulation is performed in Matlab® R2013a Simulink® tools. The S-function block is used to build the observer model by using C-coding. The solver is set to ode1 (Euler) and the sampling period is set to 100μs. An ideal inverter and pulse width modulator are assumed which means the induction machine is fed directly from the reference voltages Fig 5.1.

5.4.1.1 Open loop operation

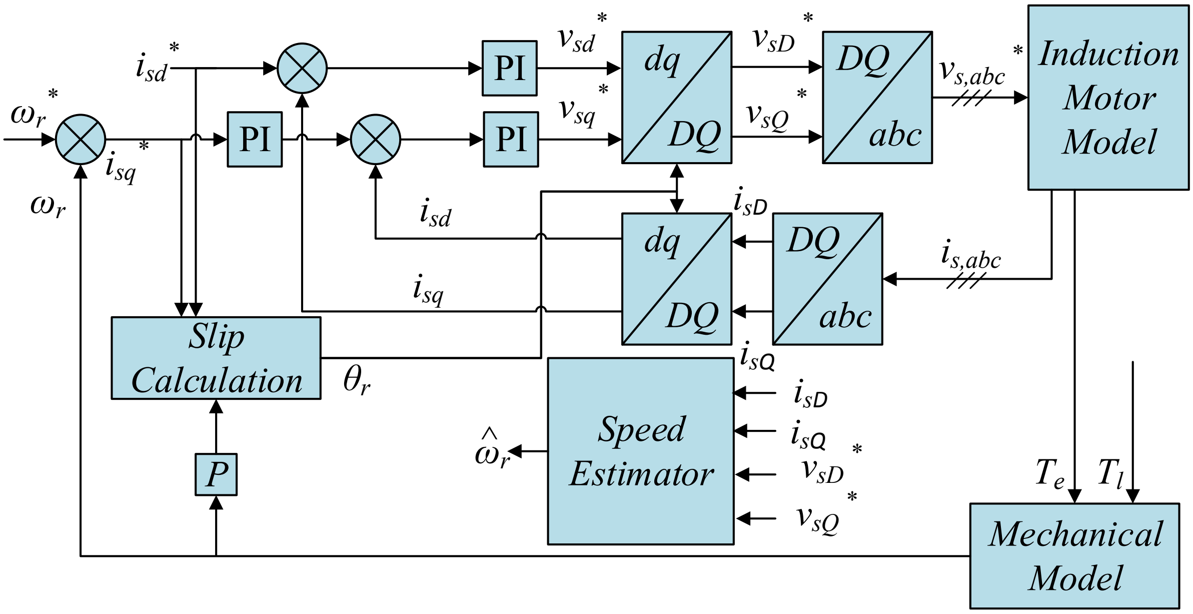
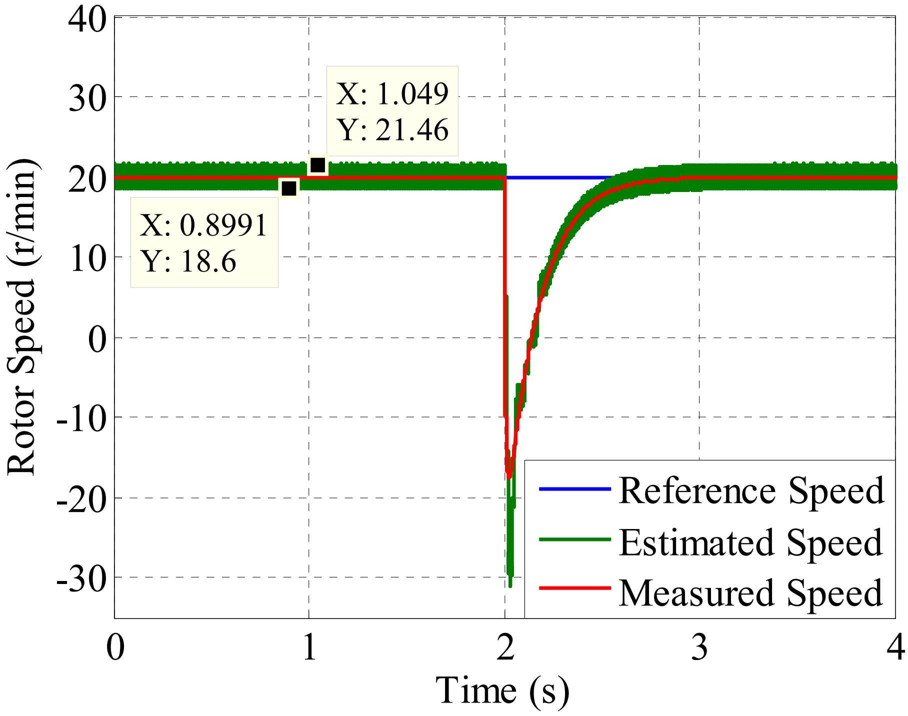


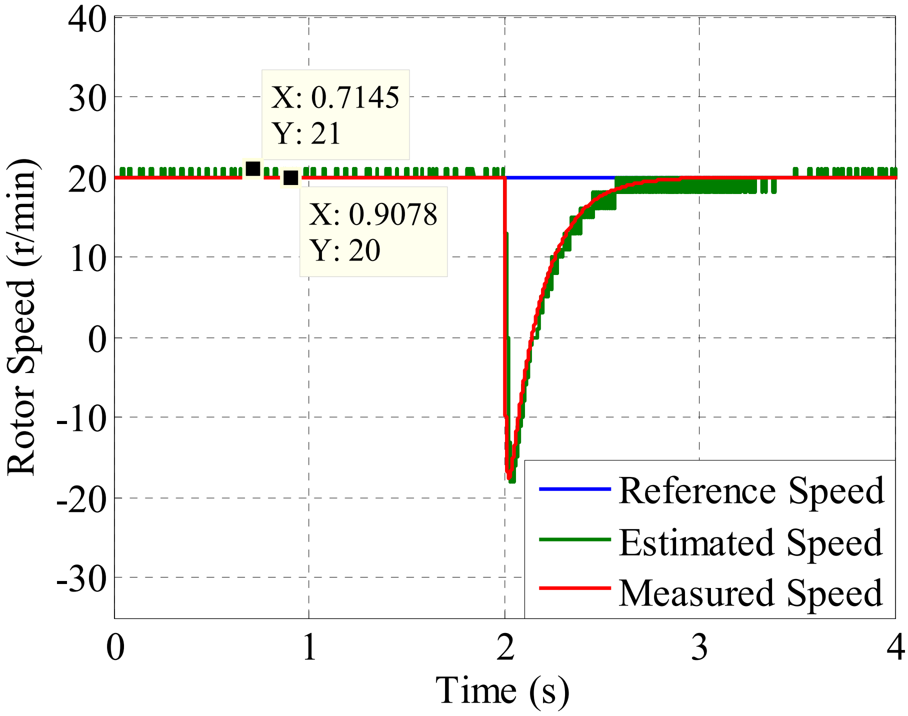
Fig. 5.2 Structure of drive system in open loop operation mode

In this operation mode the FOC scheme obtains its speed signal from the shaft encoder. The block diagram of the system model is shown in Fig 5.2, where the speed observer in the diagram can be either the classical rotor flux MRAS observer or the MP-MRAS observer depending on which of the two observer systems is applied.

In the first test, while the drive system is running at no load and 20 r/min (1.33% of the rated speed), 75% of the rated torque is suddenly applied at the time instant 2 Sec. The performance of the drive system with the two MRAS observers (the MP-MRAS and the classical MRAS observers) is shown in Fig 5.3, and Fig 5.4. The results of this test show the superiority of MP-MRAS observer in comparison with the classical algorithm. The oscillations in the case of the predictive observer vary between 20 r/min and 21 r/min which is 65.2% less than the case of the classical MRAS where the speed oscillates between 21.47 r/min and 18.6 r/min. By comparing the speed tuning signal in both cases in Fig 5.4 it can be noticed that the tuning signal in the case of the MP-MRAS is kept below 0.01 even during the transient operation. Also it reaches 0.061 in the case of the classical MRAS which is six times bigger.

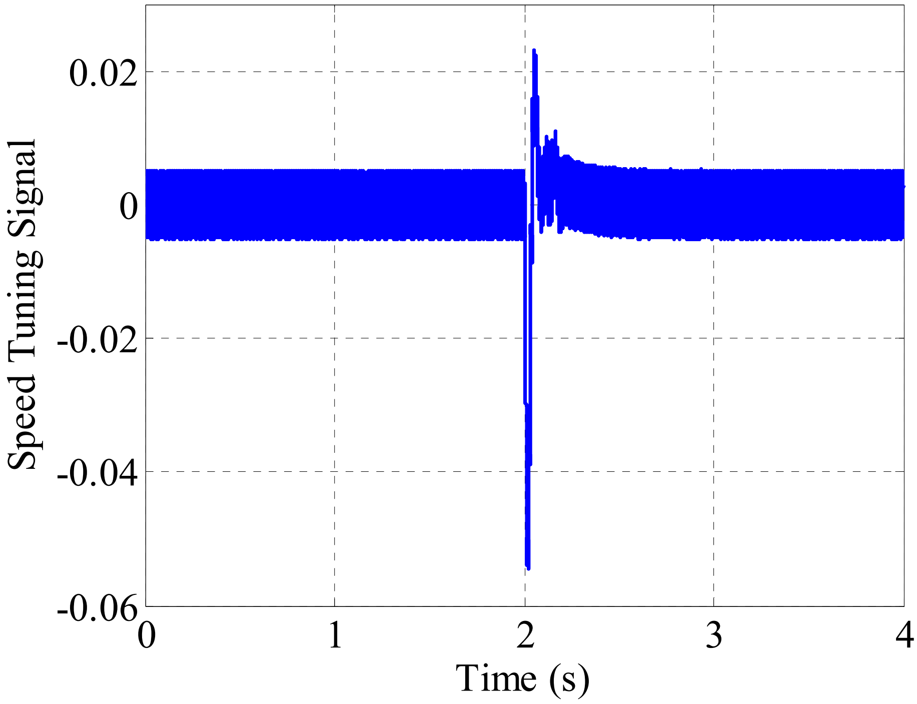


(a)

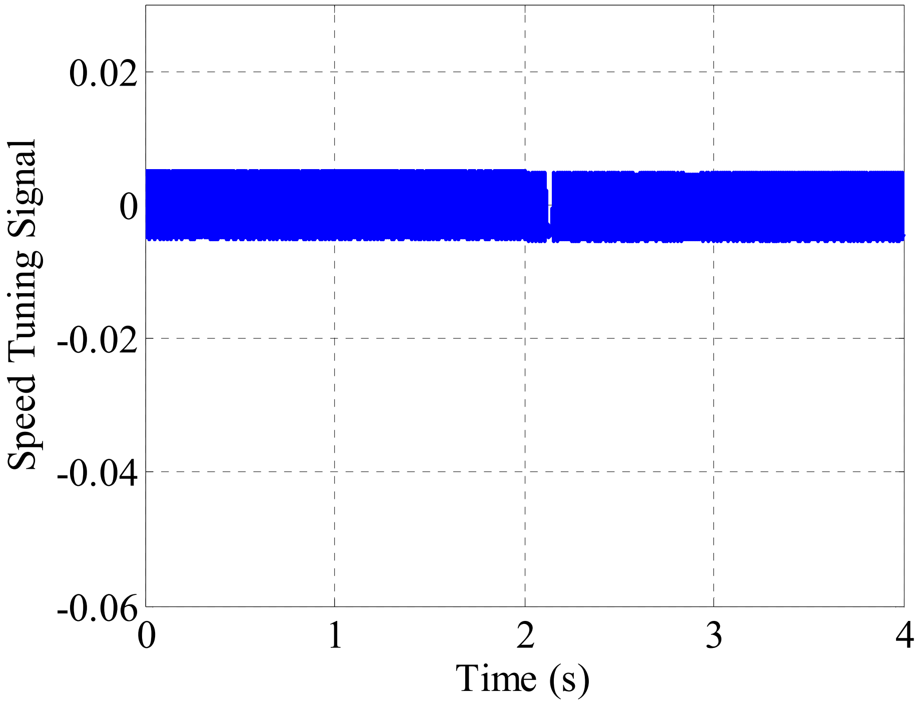


(b)

Fig. 5.3 Open loop estimation, 20 r/min and 75% load, rotor speed. (a) Classical MRAS. (b) MP-MRAS.

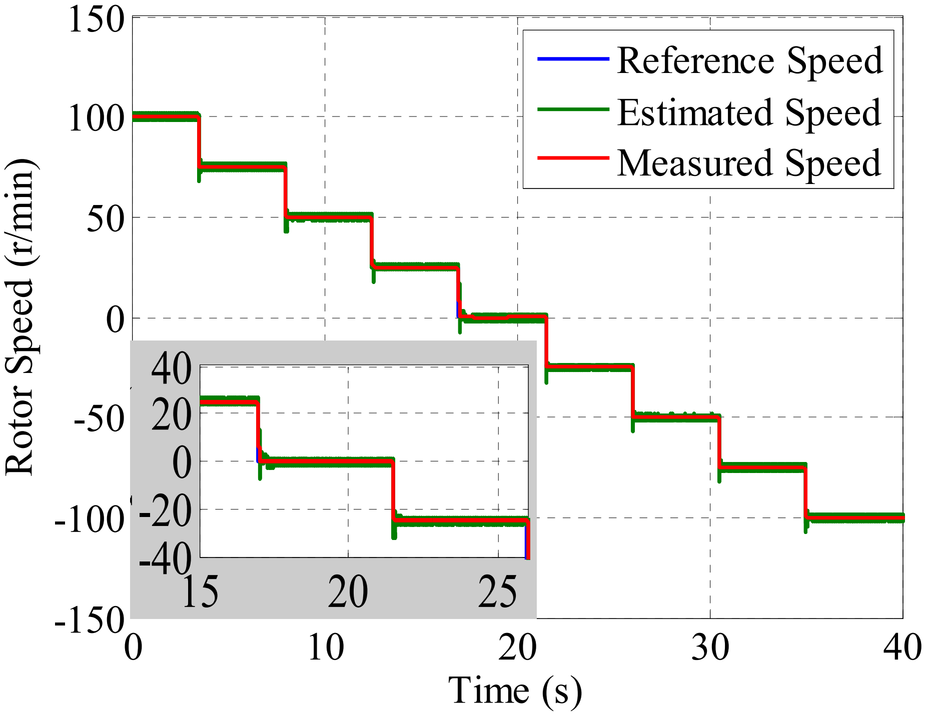


(a)

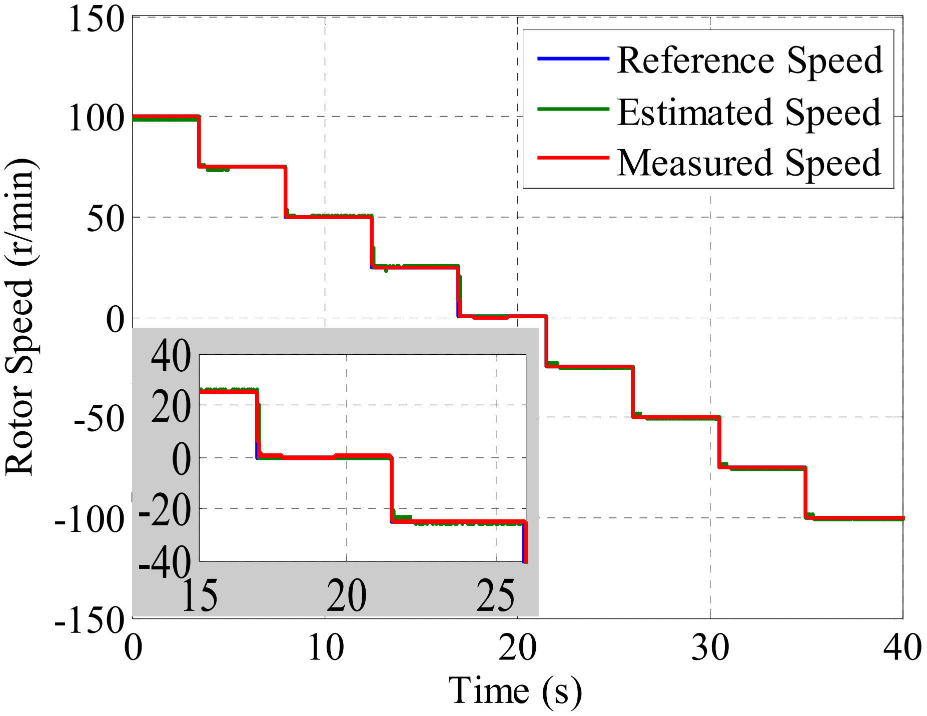


(b)

Fig. 5.4 Open loop estimation, 20r/min and 75% load, speed tuning signal. (a) Classical MRAS. (b) MP-MRAS.

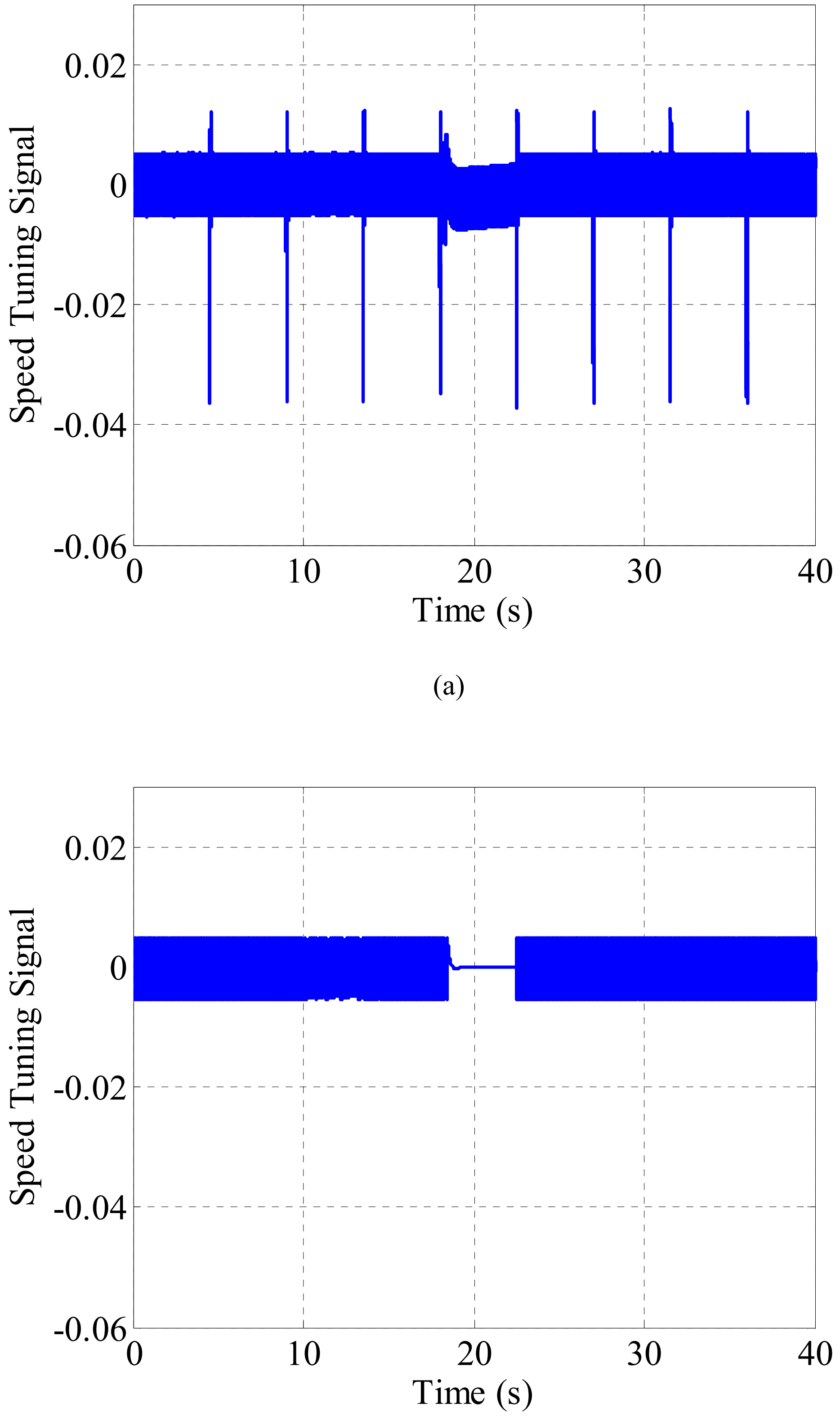


(a)



(b)

Fig. 5.5 Open loop estimation, 63% load, low-speed motoring and regenerating operation, rotor speed. (a) Classical MRAS. (b) MP-MRAS.



(a)

(b)

Fig. 5.6 Open loop estimation, 63% load, low-speed motoring and regenerating operation, speed tuning signal. (a) Classical MRAS. (b) MP-MRAS.

In the second test, the reference speed is changed in eight steps between 100 r/min (6.66%) to -100 r/min (-6.66%), including zero speed and regeneration operation, over a 40 second period of operating time. During the first half of the experiment, 63% of the rated load is applied to oppose the rotation which means that the machine is operating in the motoring mode (positive speed and positive torque), whereas the same amount of torque is applied to support the rotation over the second half which correspond to the motor operating in the regenerating mode (negative speed and positive torque).

From the results (Fig 5.5) it can be noticed that the MP-MRAS observer provides a better performance in comparison with the classical MRAS observer. The amount of the oscillation is significantly reduced over all the different operating speeds including the zero speed and the regeneration region. The speed undershoots after each speed change are also less in the MP-MRAS observer which means a better dynamic response is achieved. In terms of the speed tuning signals (Fig 5.6) it can be observed that this signal is kept under 0.01 over the whole range of speeds. This includes the transient and zero speed operation where it reaches up to 0.042 in the case of the classical MRAS observer. This means that a better tracking can be achieved between the reference and the adaptive models at all different speeds and operating conditions.

5.4.1.2 Sensorless operation

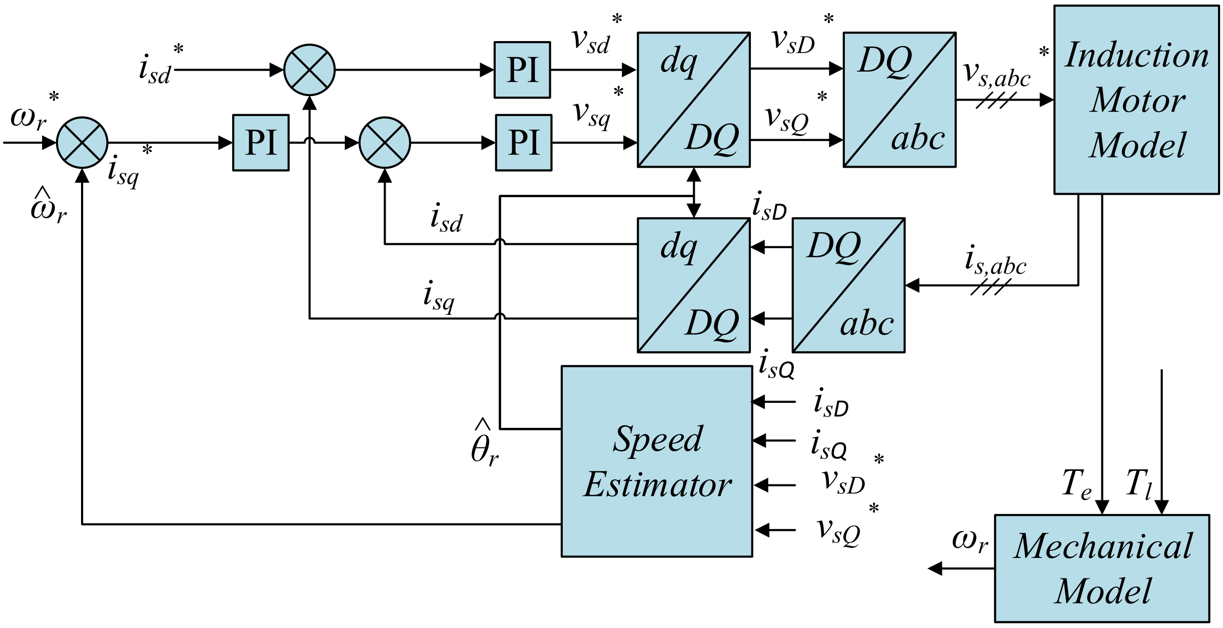


Fig. 5.7 Structure of drive system in sensorless operation mode

Fig 5.7 shows the structure of the drive system when it operates in closed loop (or sensorless) operation mode. In this operation mode, the FOC scheme is driven by the estimated speed. The performance of both MRAS observers is tested at different speeds and load conditions.

In the first test, the reference speed is set to 75 r/min, and the drive is run in the sensorless operation mode with zero mechanical load. At the time 2 sec, 75% of the rated load is suddenly applied and both the speed signal and the speed tuning signal are traced. Figs 5.8 and 5.9 show the results of the test.

The results show the superiority of the proposed observer in comparison with the classical MRAS observer in terms of the oscillation reduction in the speed signal (Fig 5.8) both before and after applying the load and during the transient operation. From Fig 5.9, it can also be noticed that the predictive observer can produce better tracking between the adaptive and the reference model by keeping the speed tuning signal as small as 0.01 at all the operating points, whereas it reaches to 0.1 in the classical observer case when the load is applied, which means 90% improvement.

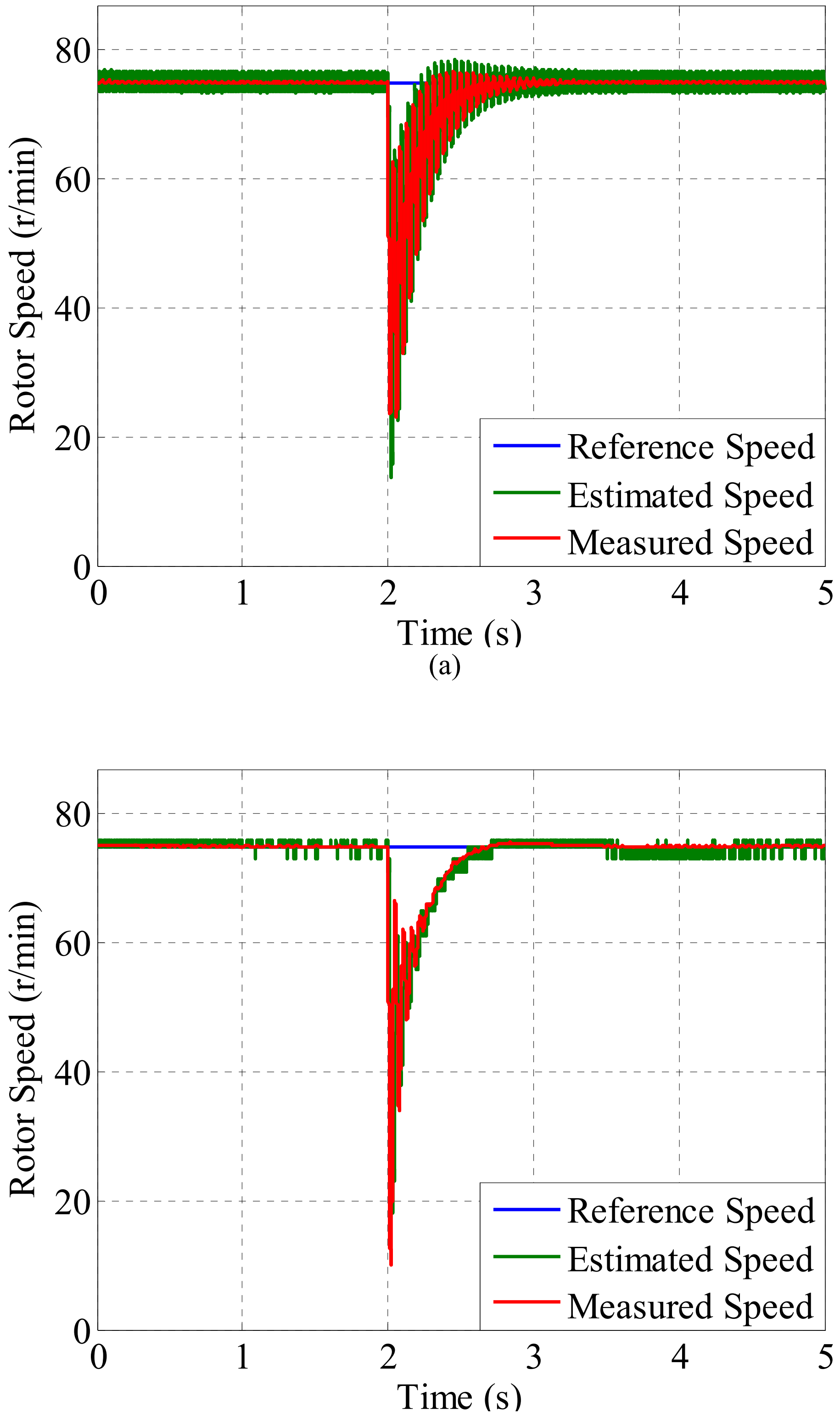
For the second test, the speed is set initially to 40 r/min (2.66% of the rated speed) and the full rated load is applied, and at 2 sec the speed reference is changed to 100 r/min (6.66% of the rated speed). Figs 5.8 and 5.9 show the speed and the speed tuning signal corresponding to the test, and once more the MP-MRAS observer system shows a better speed estimation with a reduced amount of noise in the speed signal and instant recovery of the speed tuning error after the speed transient is applied.

The system robustness against the motor parameter variation is also tested. The reference speed is set to 30 r/min (2% of the rated speed), no mechanical load is applied, and the drive is run in the sensorless operation mode.

Within the first robustness test (Fig 5.12) the drive is run under normal conditions until 1 sec, and after that the stator resistance is increased suddenly by 50% in the observer model. From the figure it can be noticed that, the system with the classical MRAS observer is more affected by the resistance variation compared to the MP-MRAS system, This shows that the speed reaches 1000 r/min for the classical MRAS after 4 seconds compared to 100 r/min for the MP-MRAS system.

In the second test, the same procedure is repeated for the mutual inductance variations, where the inductance is increased by 20% after 1 second. Fig 5.13 presents the result of this test, and shows that in the classical MRAS case, the speed starts to oscillate with increased amplitude after the disturbance is applied, until it reaches 93 r/min after 4 seconds. However, in the case of the MP-MRAS observer, the system is less affected by the inductance variation and continues to run normally after the disturbance.

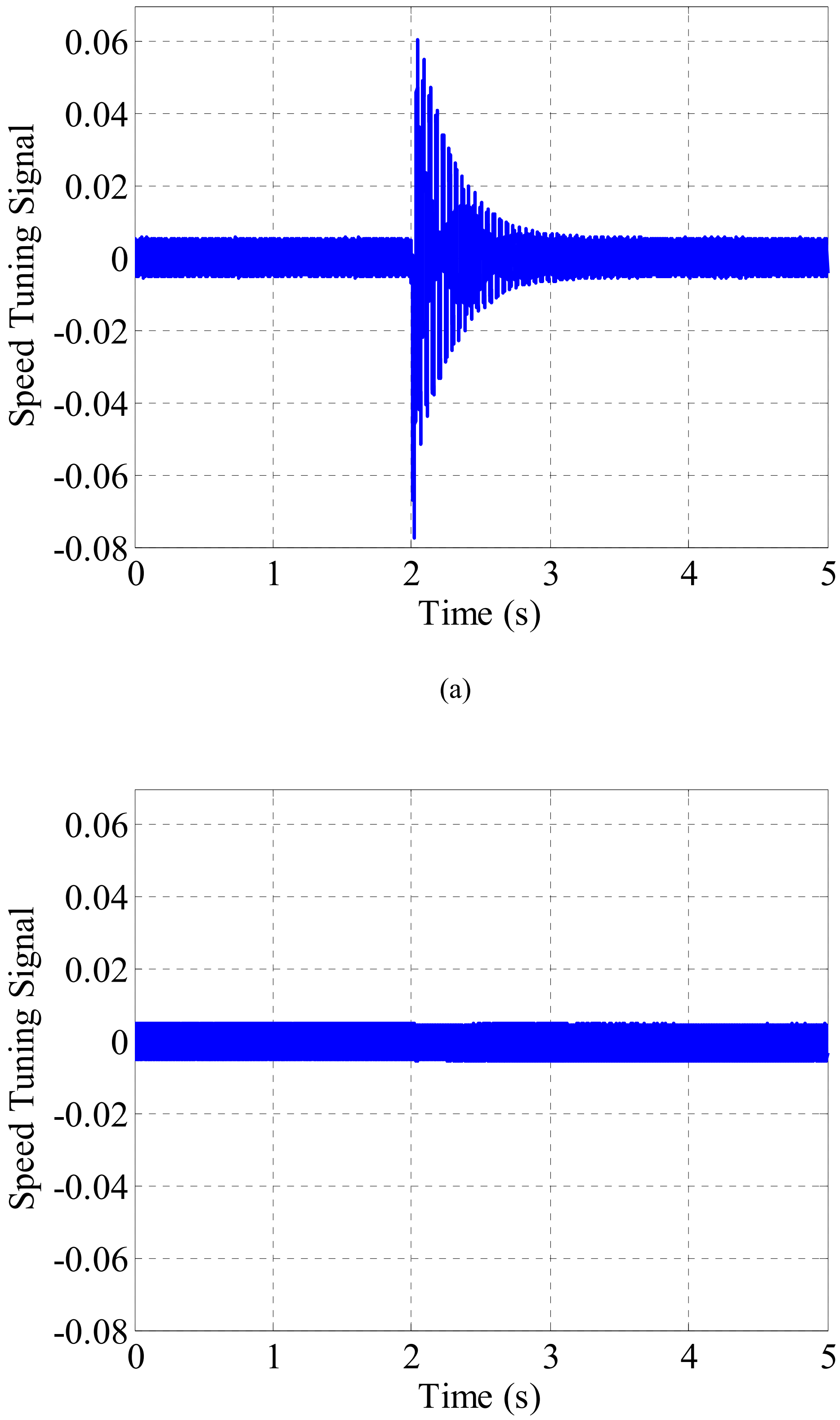
Fig 5.14 shows the result of the third robustness test where the rotor resistance is changed by 50% after 1 second. The results show that the MP-MRAS observer system is not affected by the resistance variation, and it continues to run normally after the disturbance. The case is opposite in the classical MRAS system, where the oscillations in the speed increases as a result of the rotor resister variation.



(a)

(b)

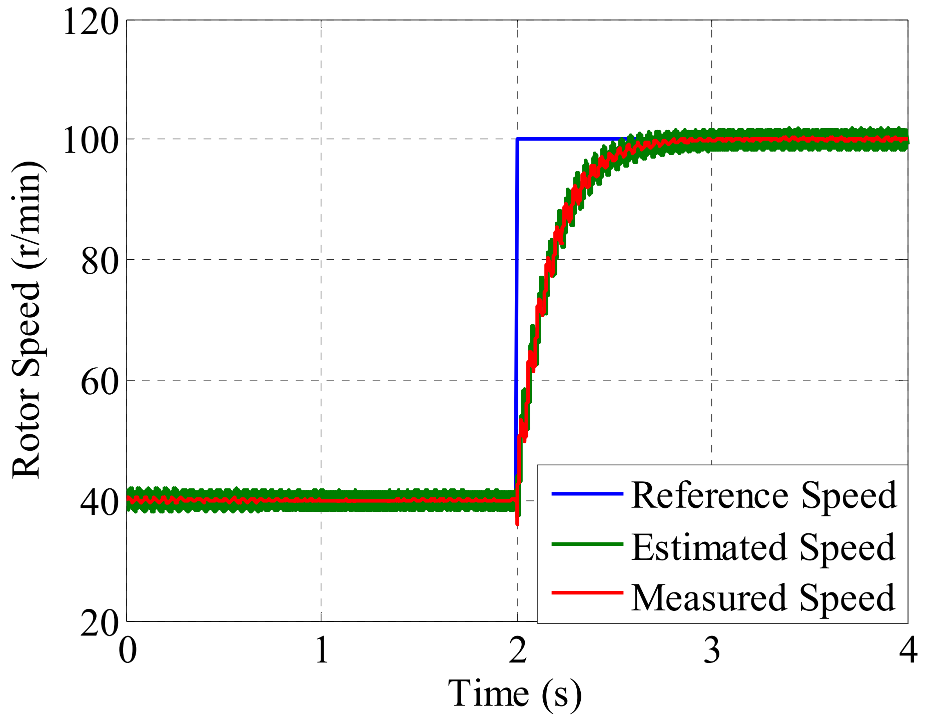
Fig. 5.8 Sensorless performance, 75 r/min and 75% load, rotor speed. (a) Classical MRAS. (b) MP-MRAS.



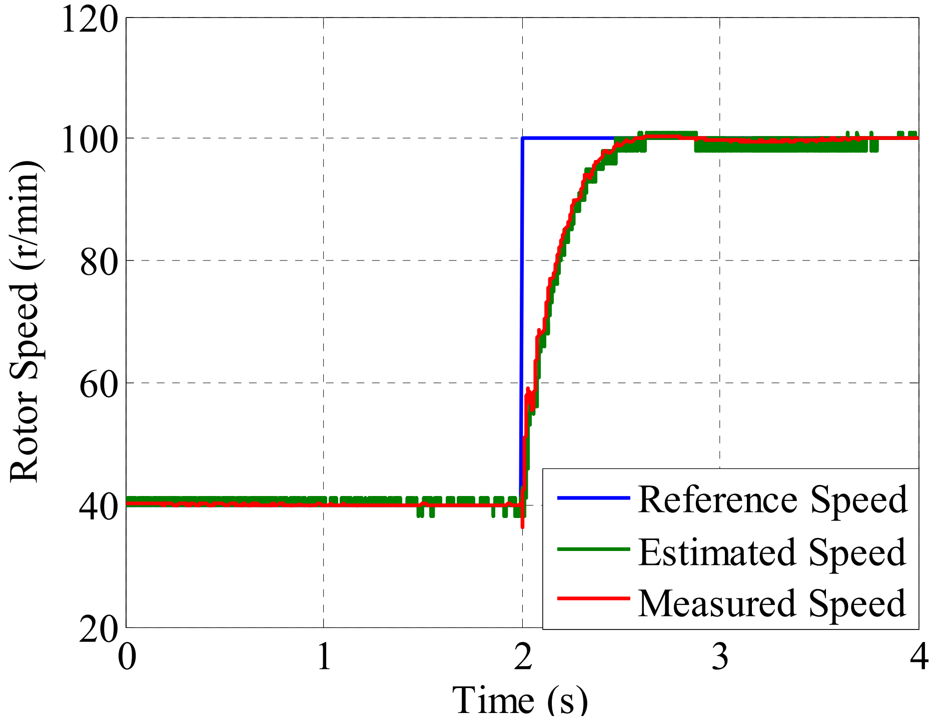
(a)

(b)

Fig. 5.9 Sensorless performance, 75 r/min and 75% load, speed tuning signal. (a) Classical MRAS. (b) MP-MRAS.

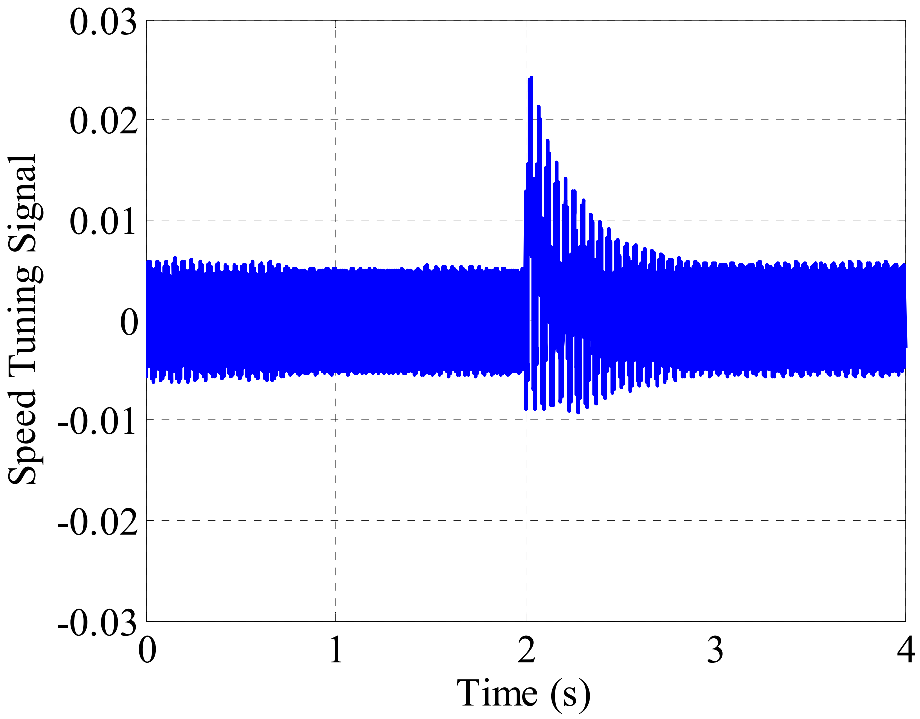


(a)

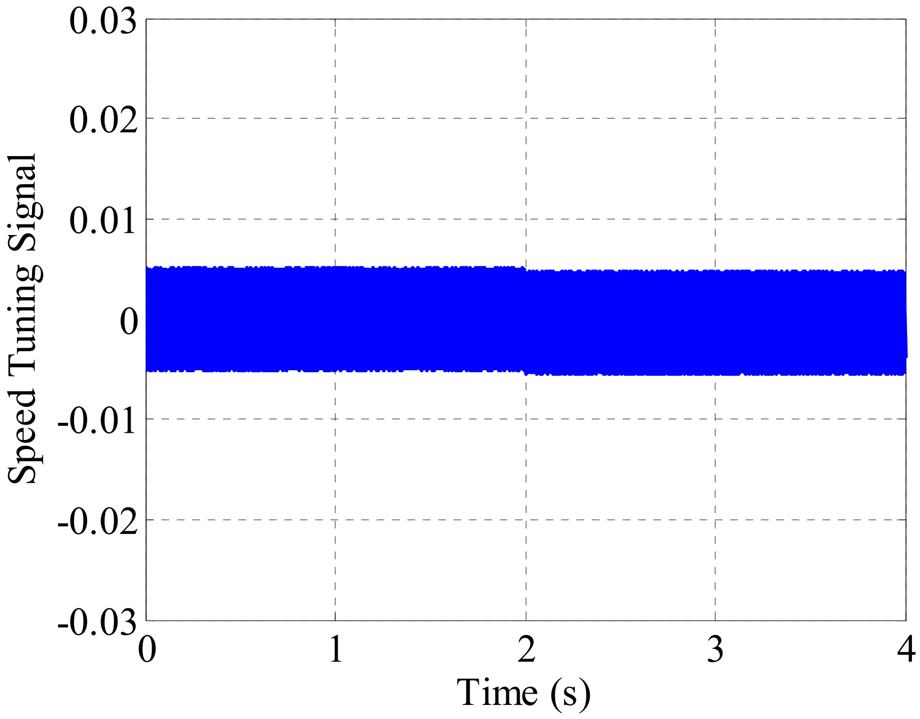


(b)

Fig. 5.10 Sensorless performance, reference speed change from 40 to 100 r/min at full load, rotor speed. (a) Classical MRAS. (b) MP-MRAS.

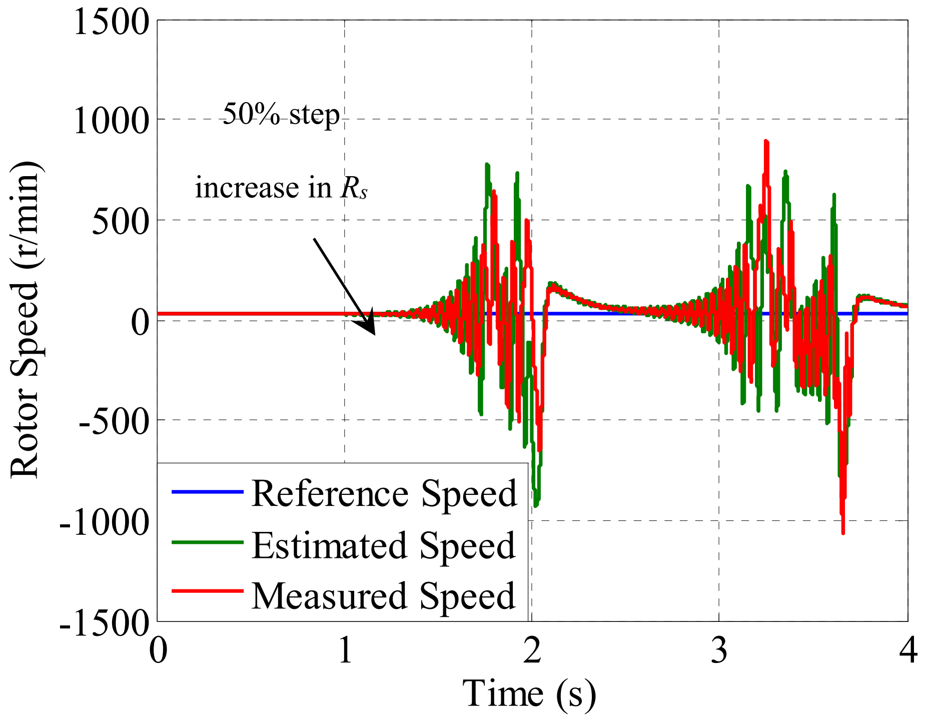


(a)

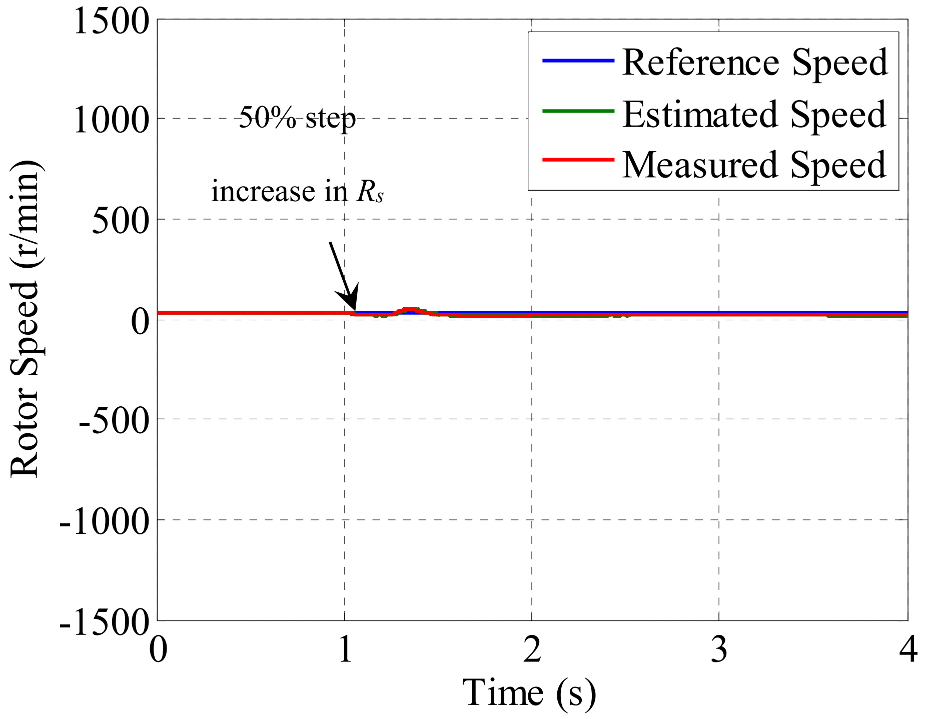


(b)

Fig. 5.11 Sensorless performance, reference speed change from 40 to 100 r/min at full load, speed tuning signal. (a) Classical MRAS. (b) MP-MRAS.

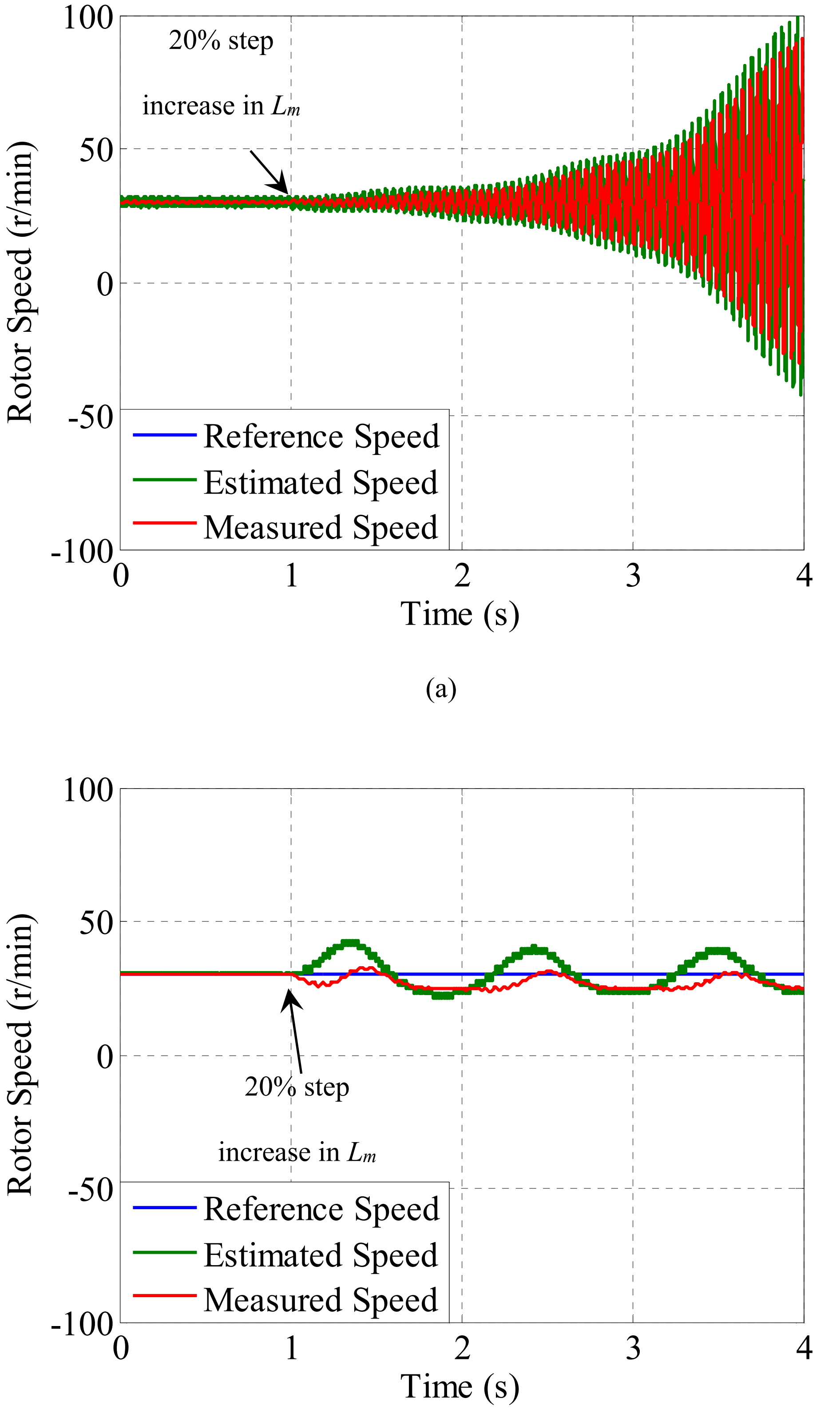


(a)



(b)

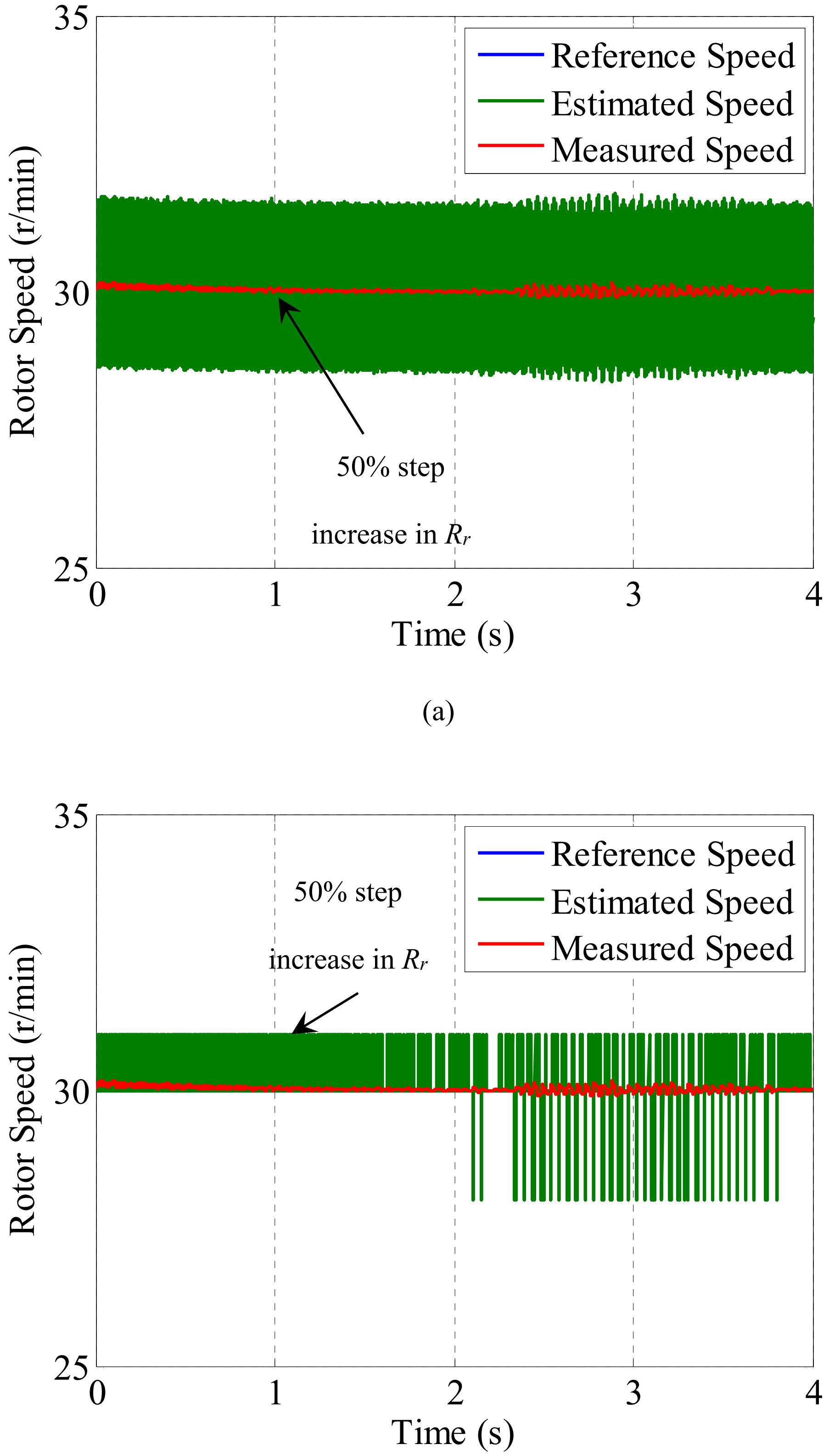
Fig. 5.12 Sensorless performance, the effect of stator resistance change (a) Classical MRAS. (b) MP-MRAS.



(a)

(b)

Fig. 5.13 Sensorless performance, the effect of mutual inductance change (a) Classical MRAS. (b) MP-MRAS.



(a)

(b)

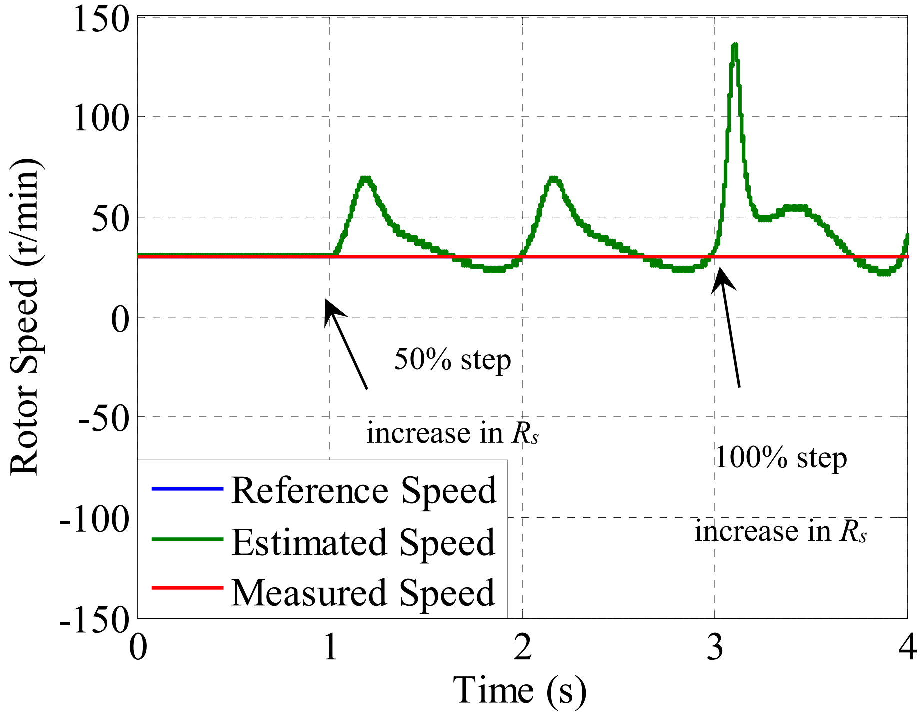
Fig. 5.14 Sensorless performance, the effect of rotor resistance change, rotor speed. (a) Classical MRAS. (b) MP-MRAS.

5.4.2 Performance comparison of the MP-MRAS and the improved MP-MRAS observers

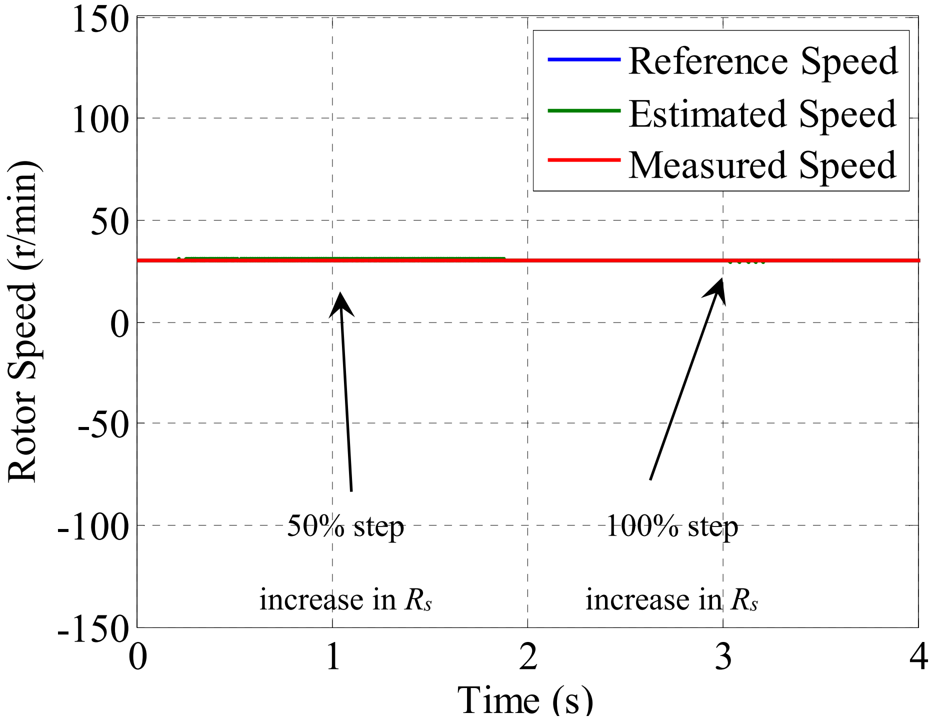
As explained in the previous chapter, the improved MP-MRAS observer is introduced to overcome the problem of the sensitivity to the motor parameter variation in the MP-MRAS observers, and the oscillations in the estimated speed at light-loading conditions. Many tests are carried out using Matlab® R2013a Simulink® tools, and the c-code S-function block.

Within the first test, the effect of R_s variation on the both the speed and flux estimation is investigated. The drive is set to sensorless operation mode, and the reference speed is set to 30 r/min. The value of the stator resistance is increased over the simulation twice, by 50% and 100% respectively. Fig 5.15 shows the speed response of the normal and improved MP-MRAS observer systems corresponding to the test. After the first stator resistance jump, the speed in the normal MP-MRAS system increases slightly and the system continues to operate in a stable manner until the second resistance change is applied where the system loses its stability and the estimated speed increases to above 1000 r/min after 2 seconds. Compared to the normal MP-MRAS, the improved observer system is barely affected by the stator resistance change and the system operates normally over the whole simulation. The same observations can also be found from rotor flux waveforms (Fig 5.16), where in the normal MP-MRAS the flux waveforms are significantly distorted by the stator resistance variations. This is in comparison to the modified MP-MRAS which successfully maintains the sinusoidal waveforms of the fluxes despite the massive change in the stator resistance.

A similar test is carried out to examine the effect of the mutual inductance variation on the speed and flux estimation. The drive is set to operate in sensorless operation mode, and the reference speed is set to 30 r/min. The value of the mutual inductance is increased twice over the simulation period, by 20% and 50% respectively. Fig 5.17 shows the speed response of the normal and improved MP-MRAS observer systems corresponding to the test. After the first mutual inductance jump, the speed in the normal MP-MRAS system starts to oscillate and the system continues to operate in a stable manner until the second jump is applied where the oscillations become very large and the system loses its stability. Compared to the normal MP-MRAS, the improved observer system is barely affected by the inductance change and the system operates normally over the whole simulation. Fig 5.18 illustrates the flux response to the inductance change and it shows that in the case of the normal MP-MRAS, the flux signal is distorted significantly after changing the mutual inductance whereas the sinusoidal waveform of the fluxes is retained in the improved MP-MRAS case.

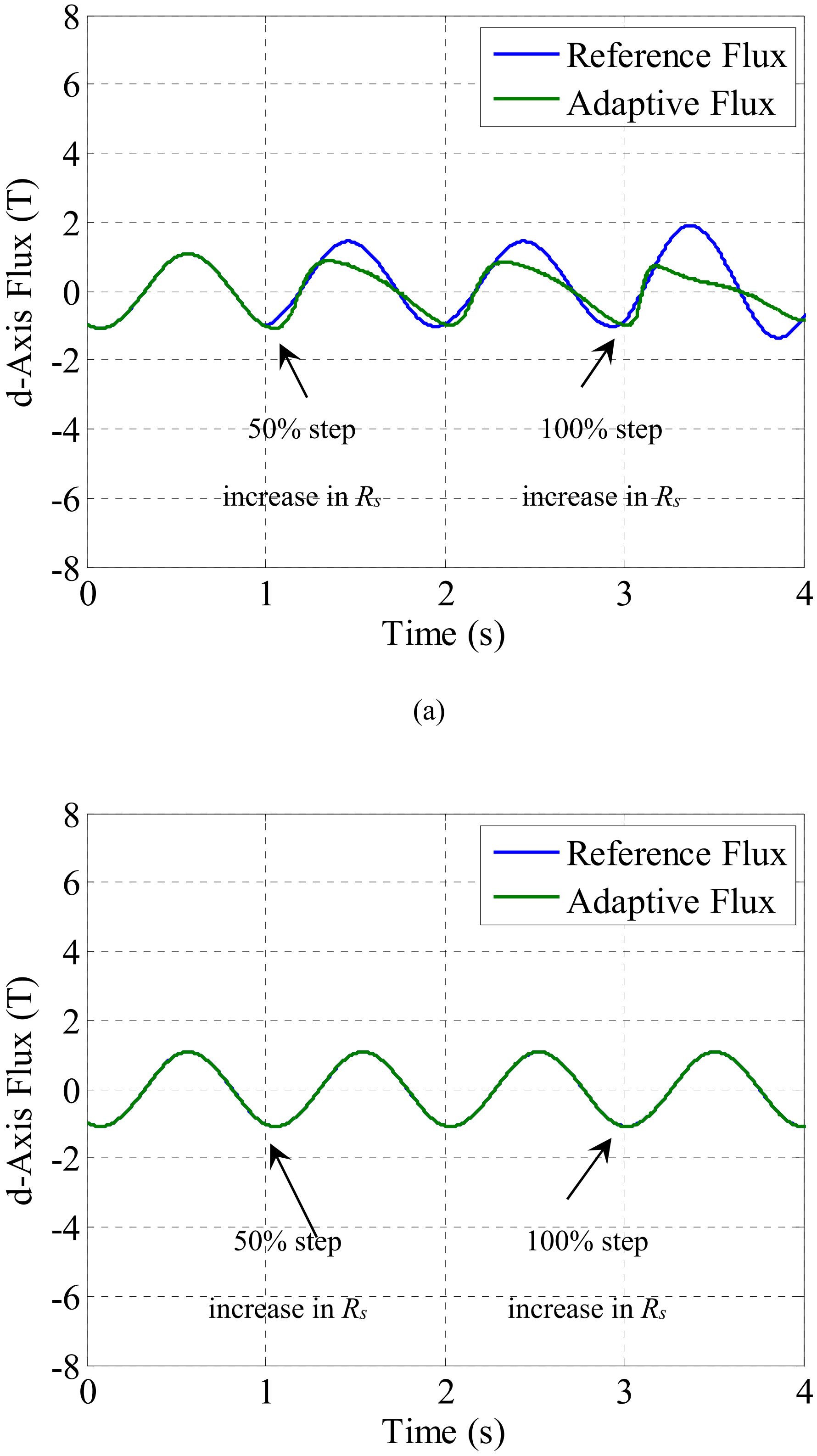


(a)



(b)

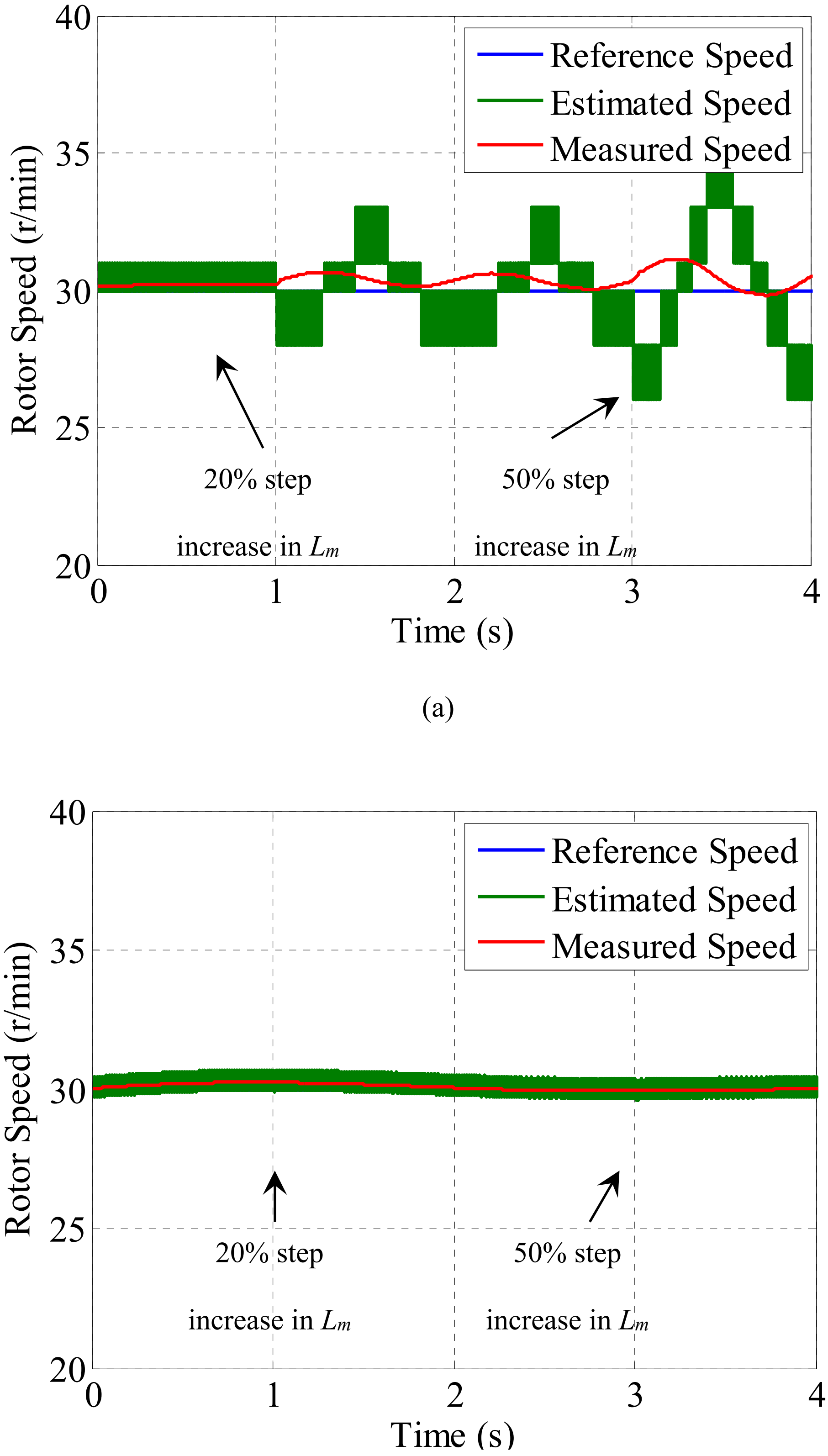
Fig. 5.15 Sensorless performance, the effect of stator resistance change, rotor speed. (a) Normal MP-MRAS. (b) Improved MP-MRAS.



(a)

(b)

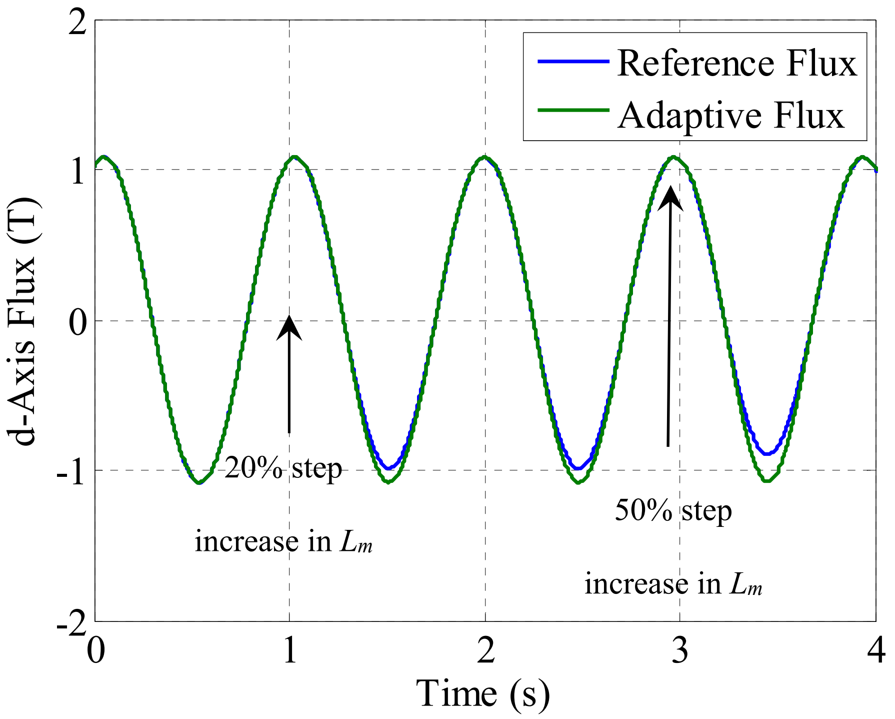
Fig. 5.16 Sensorless performance, the effect of stator resistance change, d-axis rotor flux signal. (a) Normal MP-MRAS. (b) Improved MP-MRAS.



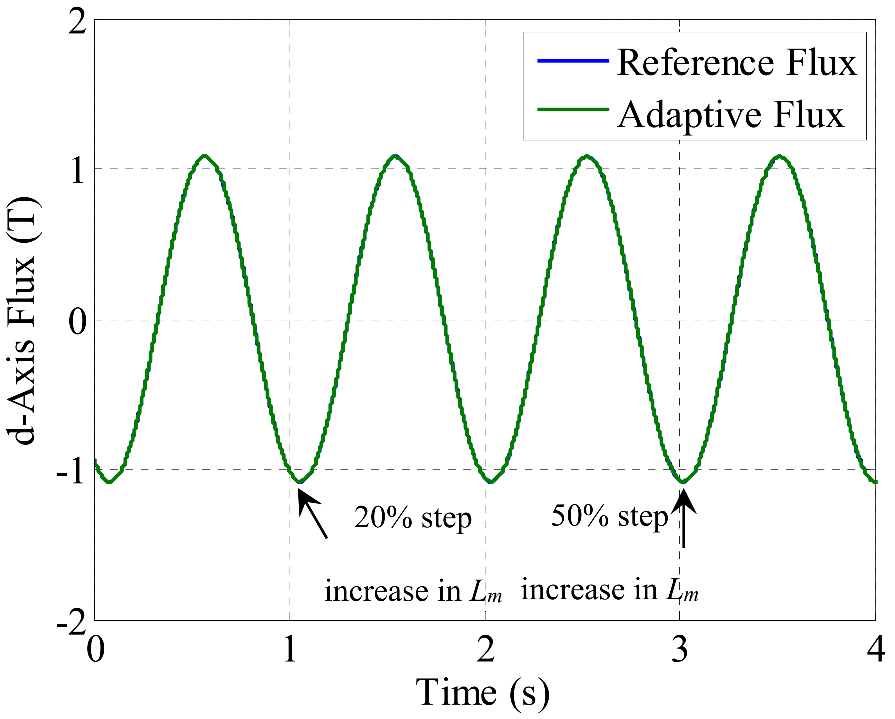
(a)

(b)

Fig. 5.17 Sensorless performance, the effect of mutual inductance change, rotor speed. (a) Normal MP-MRAS. (b) Improved MP-MRAS.



(a)



(b)

Fig. 5.18 Sensorless performance, the effect of mutual inductance change, d-axis rotor flux signal. (a) Normal MP-MRAS. (b) Improved MP-MRAS.

The results corresponding to the new speed extraction method (which was introduced in chapter IV) will be demonstrated in chapter VII (the experimental results). This problem only exists in the practical system due to the switching events in the power converter devices which is not considered in the simulation models.

5.5 Summary

In this chapter, detailed simulation tests were carried out to compare the performance of the proposed MP-MRAS and the classical rotor flux-based MRAS observer schemes. Results show a better estimation quality of the rotor speed with a significant reduction in steady-state oscillations without affecting the dynamic response as a minimum speed tuning signal is ensured in both transient and steady-state conditions. Improved robustness against motor parameter variations was also demonstrated for the proposed scheme.

In the next chapters, the experimental investigation will demonstrate the effectiveness of the proposed observer in reducing the oscillations in the estimated speed signal and improving the system robustness against the motor parameters variation, It it will also demonstrates how the proposed observer can increase the bandwidth of the speed controller and reduce the minimum operating speed of the drive system. In addition, the spectral composition of the estimated speed signals will be provided to numerically compare the amount of the oscillation in the classical and the proposed MRAS observers.

CHAPTER 6

The Experimental System

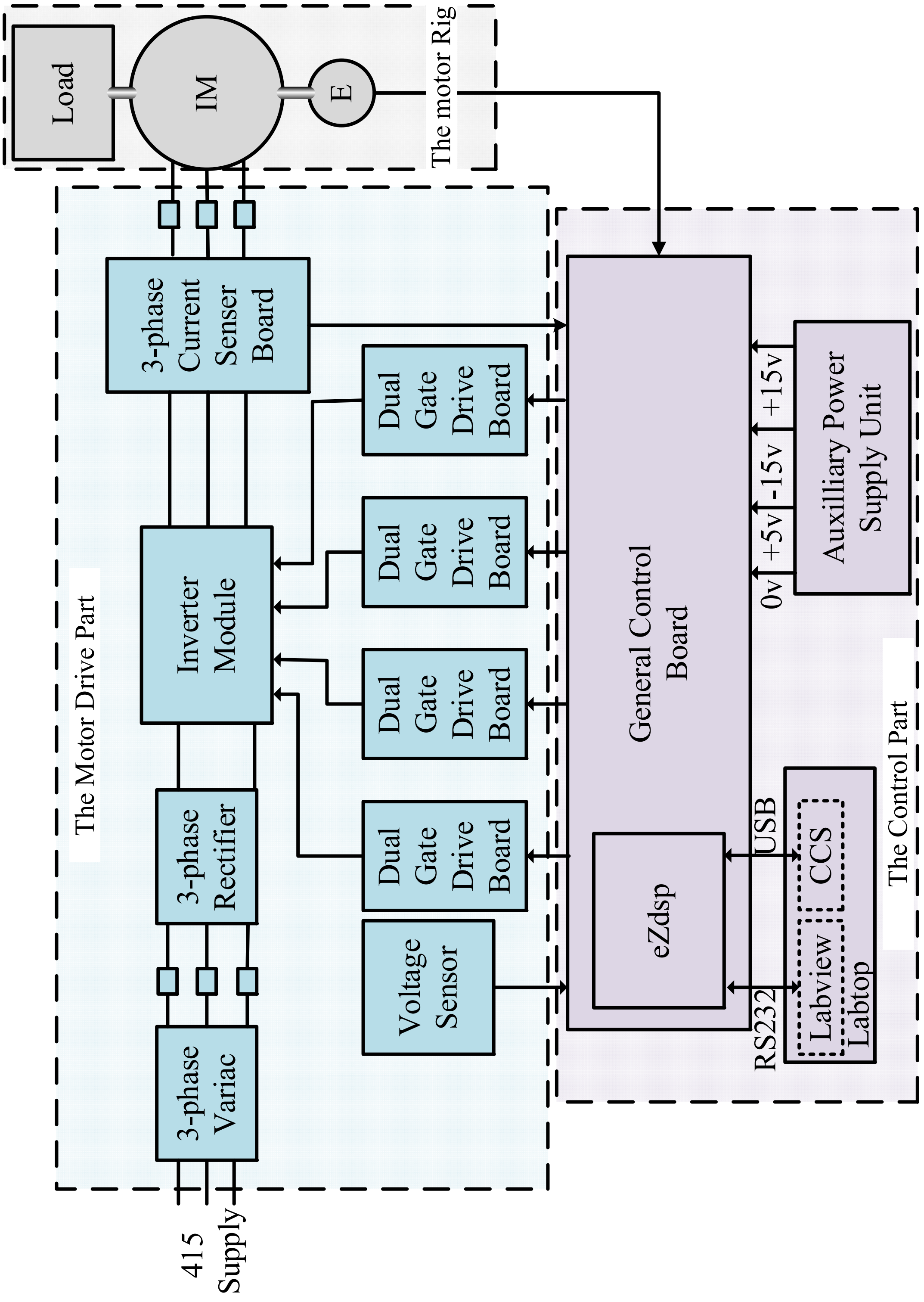


Fig. 6.1 The experimental rig block diagram

6.1 Introduction

In order to experimentally evaluate the proposed observer systems, an experimental platform, shown in Fig. 6.1, is used. This platform is based on a 2.2 *kW* squirrel cage induction machine and an eZdsp F28335 controller board. The platform setup is designed to allow both the open loop and sensorless (close loop) modes of operation.

The experimental hardware (Fig 6.2) can be separated into three different parts. The first part is the motor rig which consists of a squirrel cage IM, a PM motor-based dynamometer unit and an incremental encoder. The second part is the motor drive unit, which includes a power converter and is responsible for delivering the IM with the required power. Finally the third part is the control unit which is based on an eZdsp F28335 development board and which employs the TMS320F28335 floating point Texas Instrument microcontroller.

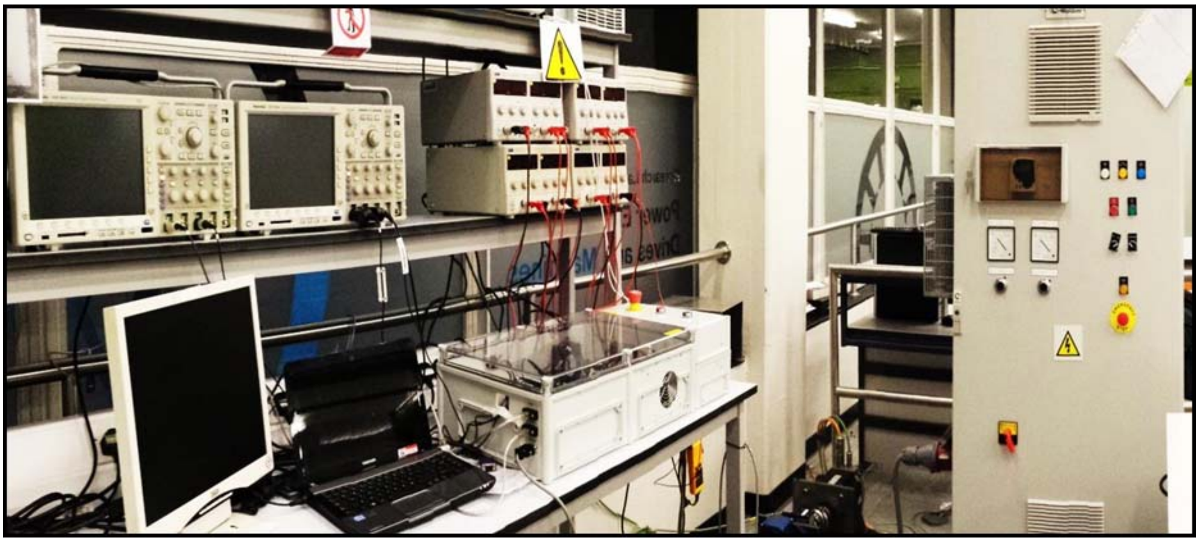


Fig. 6.2 Laboratory photograph of the experimental hardware

6.2 The Motor Rig Part

As mentioned above, the motor rig part consists of a squirrel-cage IM, a dynamometer loading unit and an incremental encoder.

6.2.1 Squirrel-cage induction motor

A 2.2kW, 380V, 50 Hz, star-connected, 4-pole, three-phase squirrel cage induction motor, manufactured by AmTecs is used as the experimental machine. The machine equivalent circuit parameters are given in table 6.1. These parameters are obtained by applying three different experimental tests: The DC test, the locked rotor test and the no-load. These tests are explained in [96].

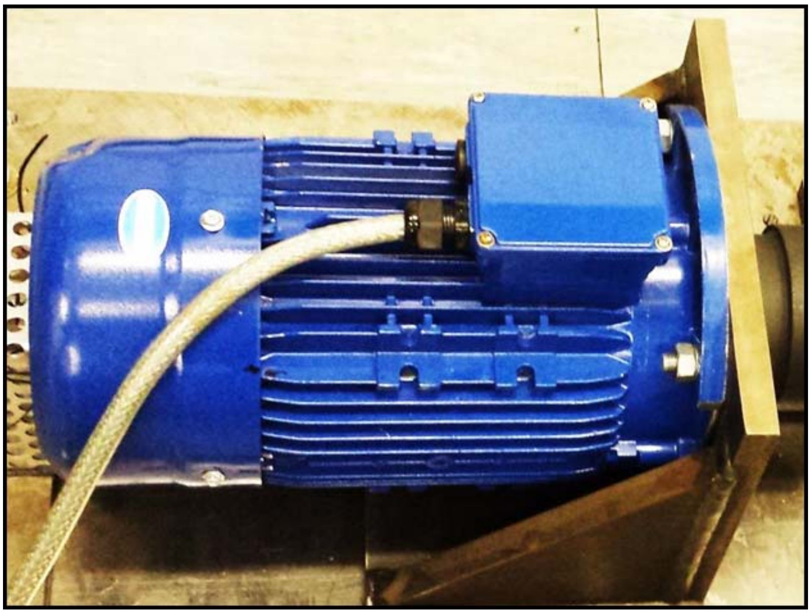


Fig. 6.3 Laboratory photograph of the squirrel-cage IM

The DC test is applied to find the stator resistance, while the locked rotor test is applied to find the rotor resistance, leakage inductance and the stator leakage inductance. Finally, the no-load test is applied to find the mutual inductance.

Symbol	Quantity	Value
R_s	Stator Resistance	2.35 Ω
R_r	Rotor Resistance	1.05 Ω
L_s	Stator Inductance	0.344209 H

L_r	Rotor Inductance	0.348197 H
L_m	Mutual Inductance	0.332090 H
J	Momentum of inertia	$0.0139 Kg.m^{-2}$
B	Friction coefficient	$0.001128 Nm/rad.s^{-1}$

Table 6.1 The induction machine equivalent circuit parameters

The last two rows in the table (the rotor moment of inertia and friction coefficient) represent the parameters of the mechanical model of the IM and they are obtained by applying the "run-down" test explained in [95].

6.2.2 The dynamometer loading unit

This unit allows an independent control of the load torque at the IM shaft, and consist of two components, the Permanent Magnet Synchronous Machine (PMSM), and the PMSM drive.

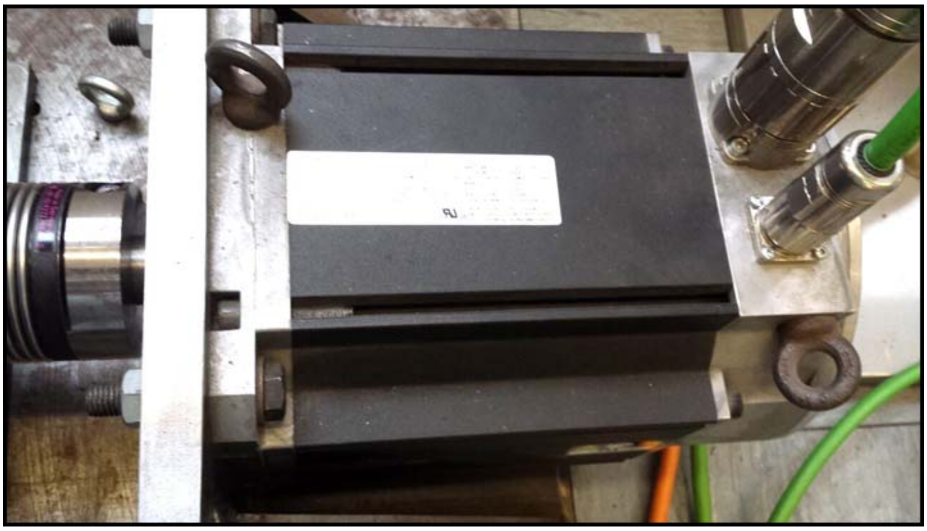


Fig. 6.4 Laboratory photograph of the PMSM

The PMSM is manufactured by Control Techniques and has the following specifications: 4.19kW, 380V, 8-pole, 20Nm and 2000RPM. Fig. 6.4 shows a Laboratory photograph of the PMSM.

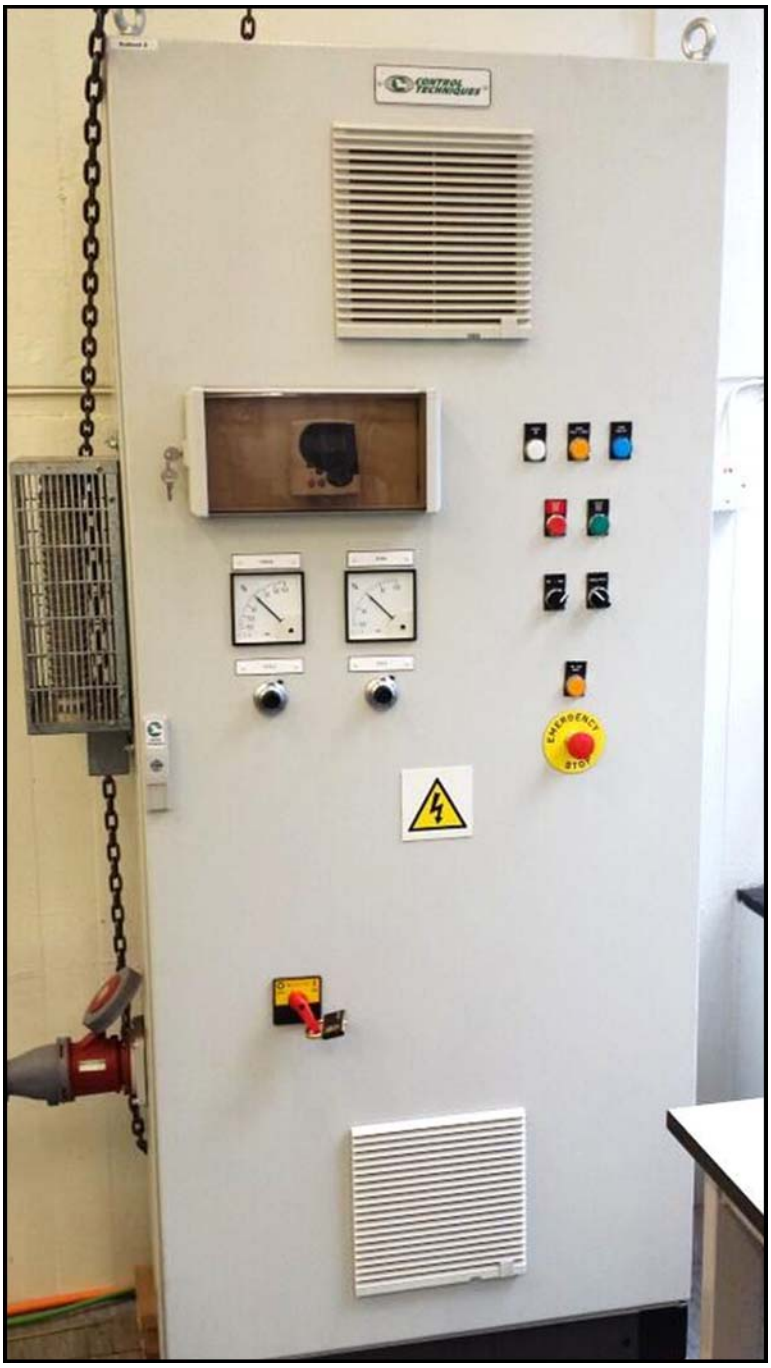


Fig. 6.5 Laboratory photograph of the PMSM.

The PMSM drive is also manufactured by Control Techniques, and it operates in two different operating modes; the normal duty mode and the heavy duty mode. Table 6.2 gives the specifications of the PMSM drive for each mode. Fig. 6.5 shows a Laboratory photograph of the PMSM drive.

Normal Duty				Heavy Duty				
Maximum output current	Nominal Power at 400v	Motor Power at 460v	Peak Current	Maximum output current	Open loop peak current	Closed loop peak current	Nominal Power at 400v	Motor Power at 460v
11 A	5.5 kW	5.5 kW	12.1 A	9.5 A	14.2 A	16.6 A	4 kW	3.7 kW

Table 6.2 The PMSM drive specifications

6.2.3 The incremental encoder

A Gurley R120 incremental encoder is used to measure the shaft speed and the position of the IM. The encoder specifications are summarized in table 6.3 and a photograph of the encoder is shown in Fig. 6.6.

Fig. 6.6 Laboratory photograph of incremental encoder

Manufacturer	Gurley Precision Instruments
Maximum line count on disc	1024
Maximum cycles /rev (quad sq waves)	16,384
Max counts/rev (after quad decode)	65,536

Internal square wave interpolation	1×, 2×, 5×, 10×, or 16×
Instrument error, ± arcminutes	4
Quadrature error, ± electrical degrees	24
Interpolation error, ± quanta	0.15
Maximum output frequency, kHz **1× square waves**	100
2× square waves	150
5× square waves	300
10×, 16× square waves	500

Table 6.3 The incremental encoder specifications

6.3 The Motor Drive Part

This part is responsible for delivering the induction machine with the required power, and it consists of a rectifier, a DC link unit and an inverter (Fig. 6.7).

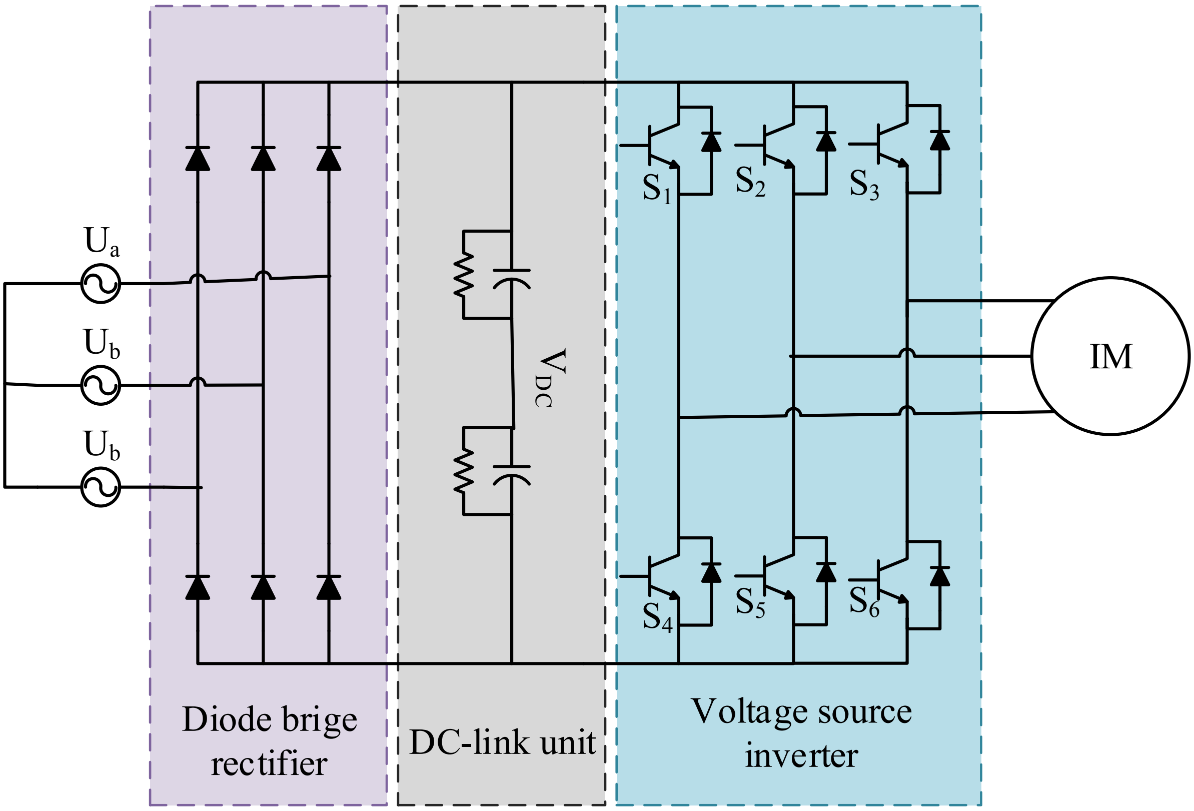
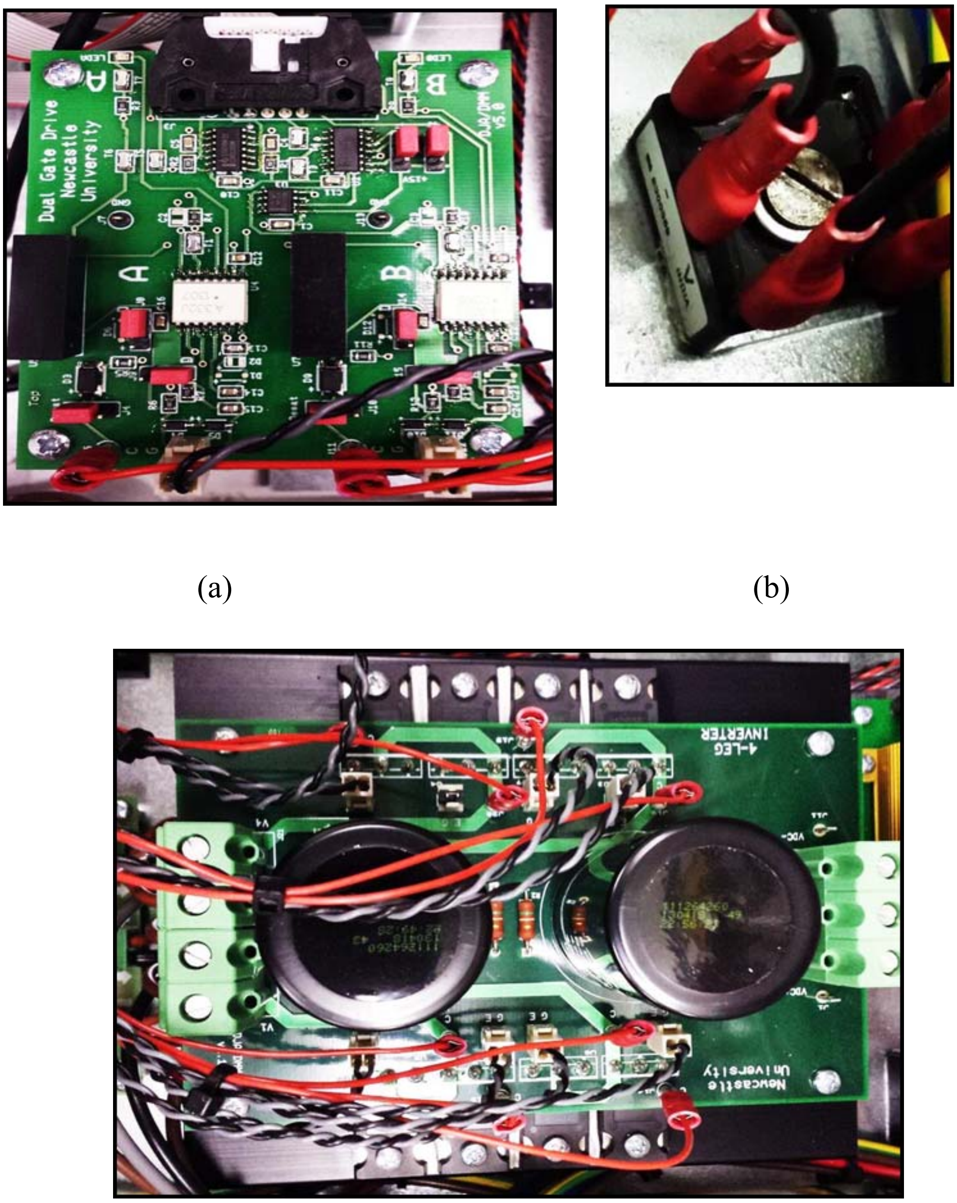


Fig. 6.7 Circuit diagram of the motor drive part

The rectifier is an *RS Components*, 3phase, 25A, 1000 V, *Vishay VS-26MT100* diode bridge which is made of six uncontrolled diodes. The output of the rectifier is connected to the DC link unit which consists of two 470 μF series connected capacitors and two 150 kΩ resistors connected in parallel with the capacitors. The aim of the capacitors is to smooth out the voltage at the output of the rectifier, and the resistors are to ensure equal voltage sharing between the capacitors. Across the DC link capacitors an LV25-P voltage sensor is connected to monitor the DC-link voltage.



(a)

(b)

(c)

Fig. 6.8 Photograph of the drive unit. (a) The gate drive circuit. (b) The rectifier. (c) The inverter and the DC-link unit.

The inverter unit consist of six IGBT switches. The switches have the following specifications: V_{CES}=1200V, $I_{NOMINAL}$=20A, $T_{J(MAX)}$=150ºC, $V_{CE(ON)}$=1.9V. The three-phase outputs of the inverter are connected to the IM terminals via three CAS-15NP hall-effect current sensors to measure the phase currents. To provide the switching signals to the power switches, six gate drive circuits are used which are based on AVAGO ACPL-332J opto-isolated driver. These drivers not only provide isolation between the power conductors and the controller, they also provide protection against short circuit currents and generate a programmable dead band between the different output signals. Fig 6.8 shows a photograph of the diode rectifier, the inverter and gate drive circuits.

6.4 The Control Part

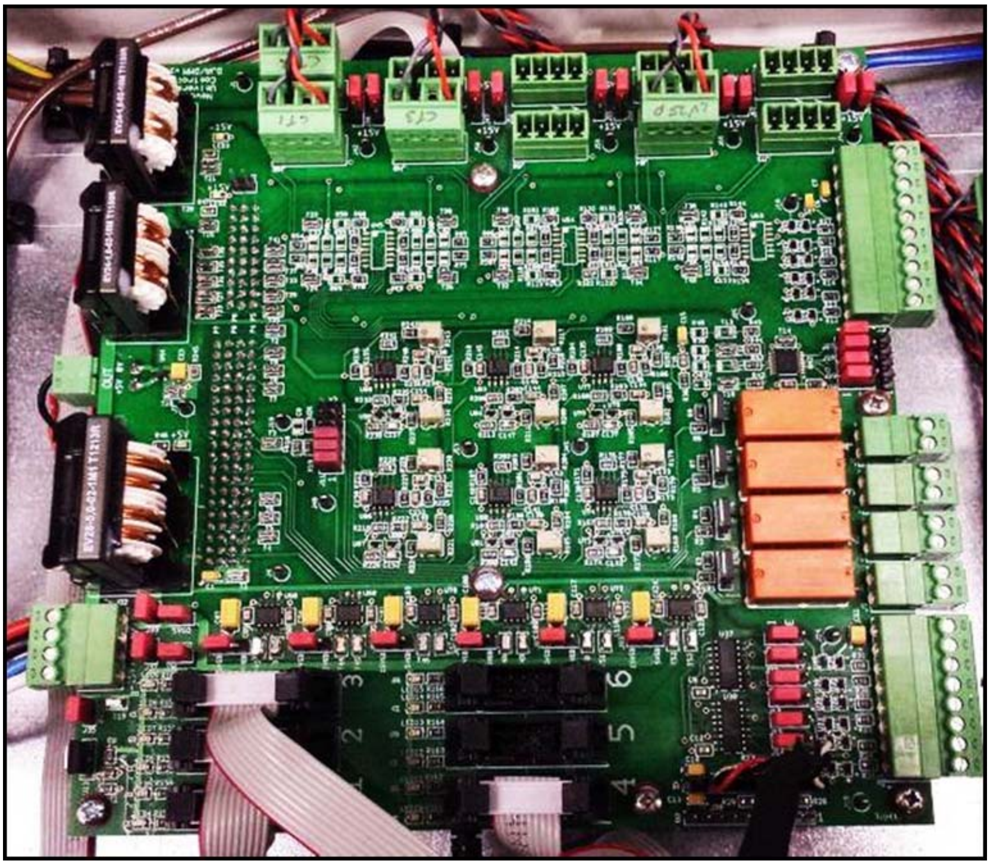


Fig. 6.9 Photograph of the general interface board

This part reads the measurements from the motor drive and establishes the control accordingly. The unit is based on an eZdsp TMS320F28335 floating point Texas Instrument microcontroller board. This controller has the advantage of high speed processing, an external SRAM memory, built-in PWM generating circuit, RS-232 interface and many other features to allow the controller meet the project requirements.

To interface the microcontroller to the motor drive circuits, an interface board designed at Newcastle University is used (Fig. 6.9). This board contains:

- Gate drive interface circuits to route the microcontroller PWM signals to the external IGBT drives.
- Sensor interface circuits to allow the different current and voltage sensor outputs to be fed into the micro controller.
- Fault detector circuits connected with the sensor interface circuits to provide fast hardware protection when the sensor readings go out of range.
- A 5604 TLV SPI Digital to Analogue converter (DAC) circuit is provided to allow use to monitor four of the internal digital signals.
- Two shaft encoder interfaces to allow the microcontroller to read the position signals of a connected encoder.

6.5 Summary

In this chapter, the experimental setup used in validating the proposed observer scheme has been presented. The hardware has be divided into three different parts, the motor rig which is based on a 2.2 *kW* squirrel cage IM, the motor drive unit which consists of the power converter and was responsible for delivering the power to the IM, and finally, the control unit which is based on an eZdsp F28335 board.

In the next chapter, the experimental results will be presented which will validate the proposed observer utilizing the experimental hardware built in this chapter.

CHAPTER 7

The Experimental Results

7.1 Introduction

To evaluate the comparative performance of the proposed MP-MRAS observers and the classical rotor flux-based MRAS scheme, extensive tests, in both open-loop and sensorless operation modes, are carried out using FOC scheme as the IM control strategy. The switching frequency of the inverter is set to 10 *kHz* and the controller-observer algorithm is operated at the same sampling frequency (10 *kHz)*, synchronised with the PWM carrier signal.

In the reference model, to minimize drift and initial condition problems associated with pure integration, the integrator is replaced by a low pass filter with a cut-off frequency of 2 Hz. The cut-off frequency of the filter should be selected to be as small as possible in order to remove the DC component from the flux signals. In addition, as the modulation signals available in the controller unit are used as voltage inputs to the reference model (3.59), a compensation for the inverter nonlinearity [3] and a dead-band compensator [71] are implemented.

Finally, it should be mentioned here that, to minimise the execution time of the proposed control system, which makes use of sine and cosine functions in the reference frame transformations, a look-up table for one period is set up of sine and cosine to replace the standard sine and cosine functions. The look-up table can execute much faster compared to the implementation of sine and cosine functions within a high level language.

7.2 Performance Comparison of the Classical MRAS and the MP-MRAS Observers

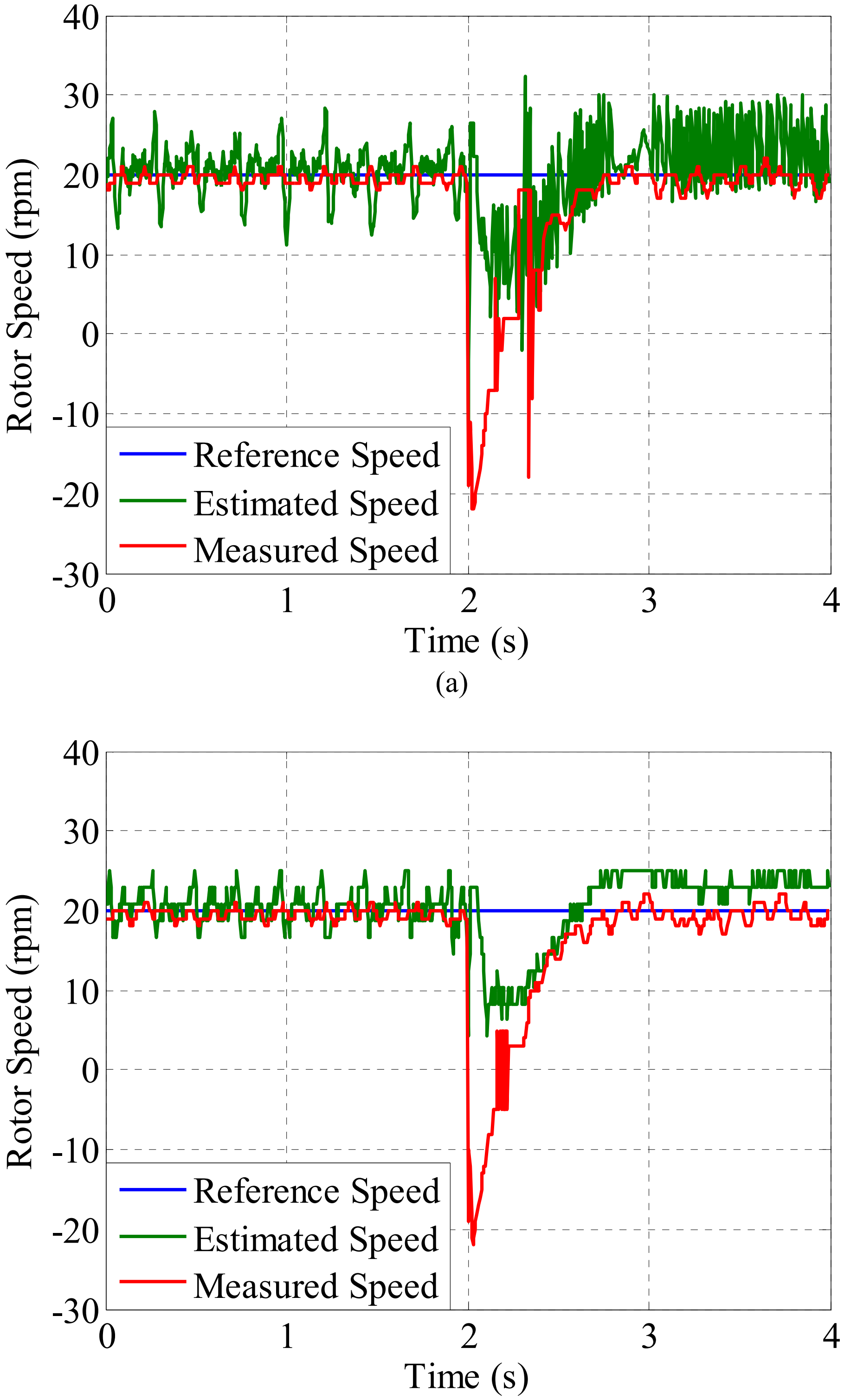
7.2.1 Open loop observer operation

As mentioned in the previous chapter, during open-loop observer operation, the FOC scheme obtains its speed signal from the shaft encoder. The PI controller gains of the classical rotor flux-based MRAS are set to K_p = 300 and K_i = 8000 which are tuned using trial-error method to obtain the optimal dynamic performance.

Figs. 7.1-7.3 show the classical and MP-MRAS observer performance for 75% load rejection at 1.33% of the rated speed (20 r/min). The MP-MRAS shows superiority in comparison with the classical MRAS. The oscillation in the estimated speed is reduced significantly and the speed tuning signal is kept below 0.02 even during the transient operation whereas it reaches 0.1 in the classical MRAS, which is five times greater. This means that the predictive observer provides better tracking between the reference and the adaptive models at all the different operation conditions. The frequency spectrum of the estimated rotor speed signal shows that the harmonic content has been reduced significantly in the MP-MRAS observer. For example, the 17 *Hz* component has been reduced from 0.028 per unit to 0.009 per unit, which is a reduction of 67.8%.

Figs.7.4- 7.6 show the classical and MP-MRAS performance when 63% of the rated load is applied and the speed reference is changed from 6.6% (100rpm) to -6.6% (-100 rpm) of the rated speed in 8 steps, including zero speed and regeneration operation. During the first half of the experiment, the load is applied to oppose the rotation which means that the machine is operating in the motoring mode (positive speed and positive torque), whereas the torque is supporting the speed over the second half and the motor is operating in the regenerating mode (negative speed and positive torque). From the results it can be seen that the predictive observer can produce speed estimation with a better quality in terms of reduced oscillations at all the different speeds including zero speed. The speed tuning signal remains less than 0.011 during all transient and steady state conditions, while it reaches 0.055 in the classical MRAS. During the regeneration region the MP-MRAS observer provides a better performance with less steady state error and oscillations.

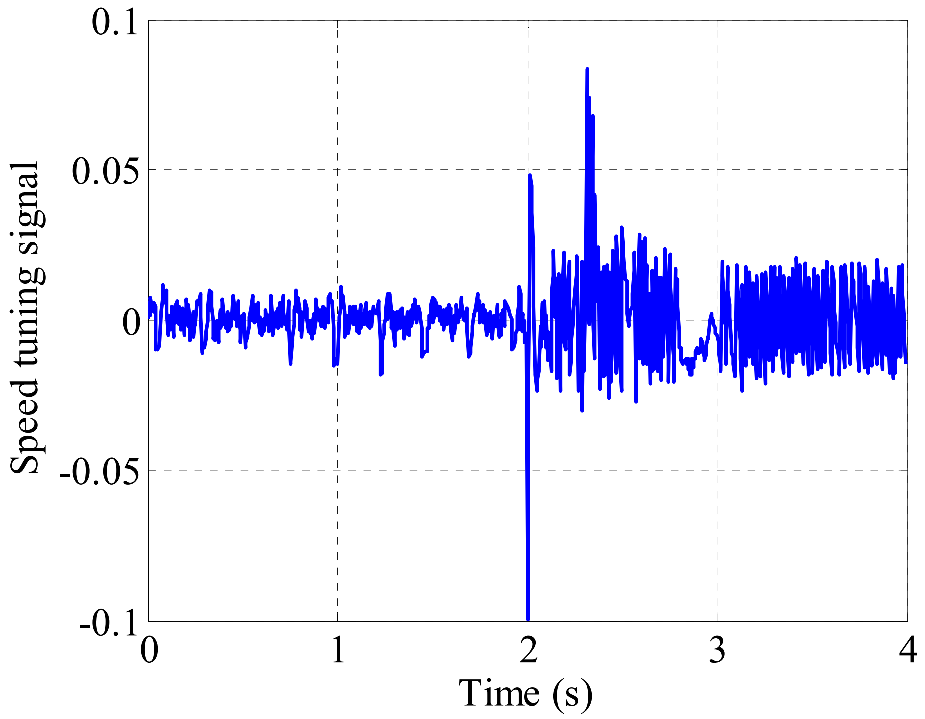
To test the proposed scheme's robustness against motor parameter variations, two experimental tests are carried out. Within the first test, Fig. 7.6 .a, a 50% step change has been applied to the rotor resistance R_r in the observer model while the machine is running at 300rpm and full load. It can be noticed from the figure that the MP-MRAS observer is less affected by the rotor resistance change with 14 r/min initial undershoot and 0.15s recovery time compared to 19 r/min initial undershoot and 0.45s recovery time for the classical MRAS. In the second test, Fig.12.b, a step change of 20% has been applied to the mutual inductance L_m in the observer model. It can be observed that the MP-MRAS scheme shows better performance with less oscillation during transients compared to the classical MRAS. In the two tests, the proposed scheme shows better steady-state rotor speed estimation with a lower noise level.



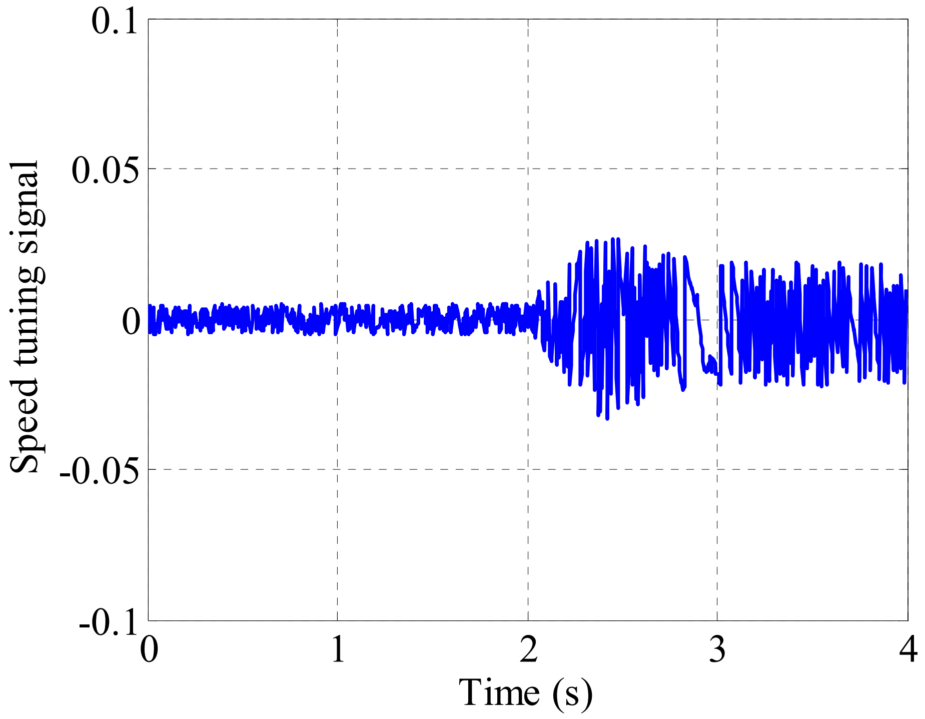
(a)

(b)

Fig. 7.1 Open loop estimation, 20 r/min and 75% load, rotor speed (a) Classical MRAS (b) MP-MRAS

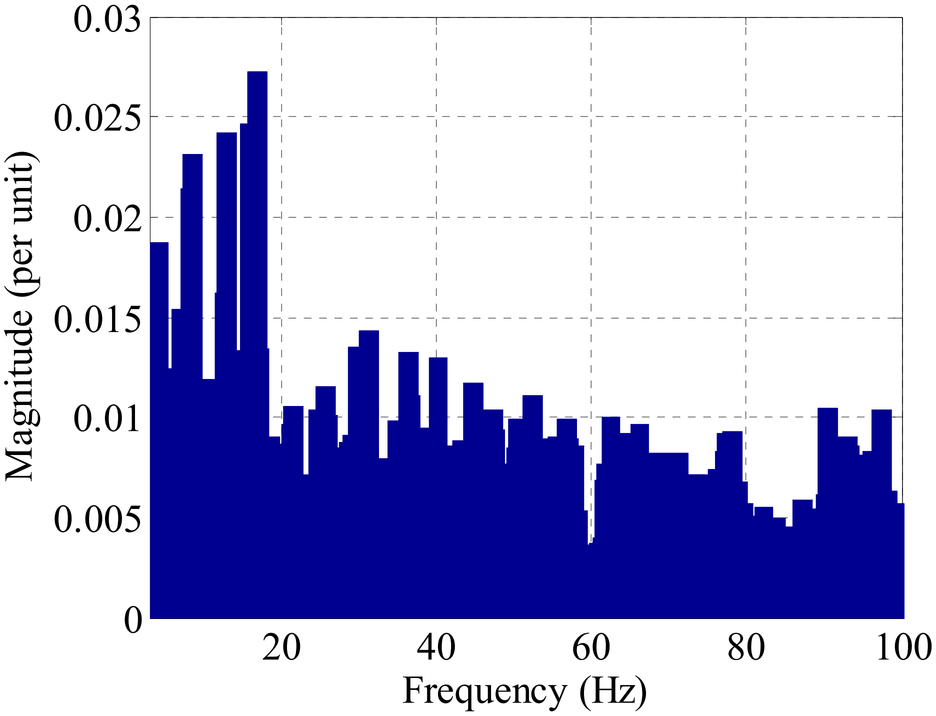


(a)

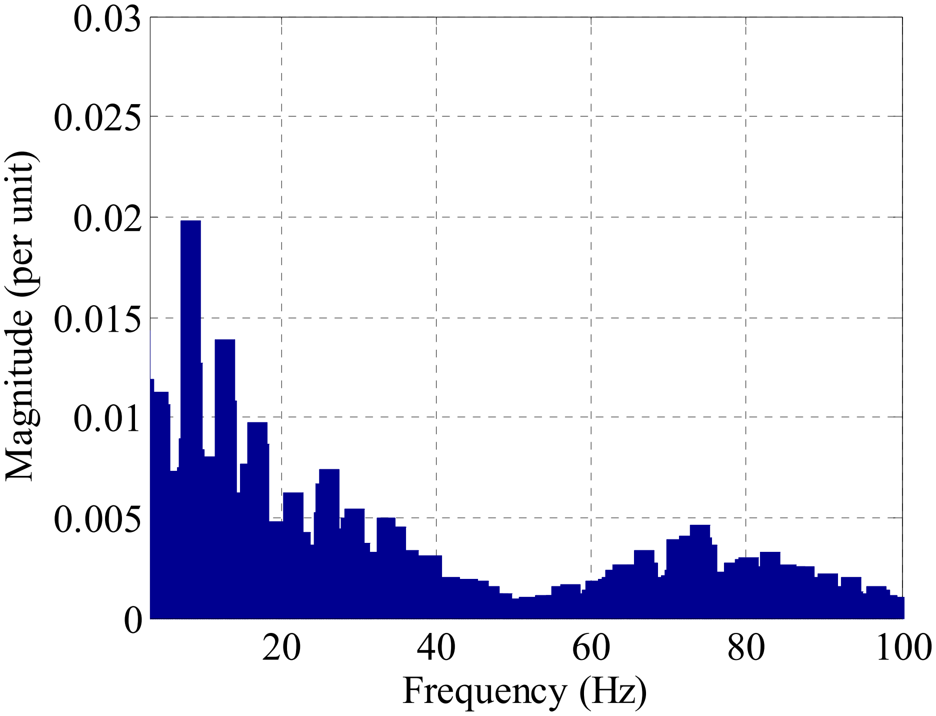


(b)

Fig. 7.2 Open loop estimation 20 r/min and 75% load, speed tuning signal (a) Classical MRAS (b) MP-MRAS

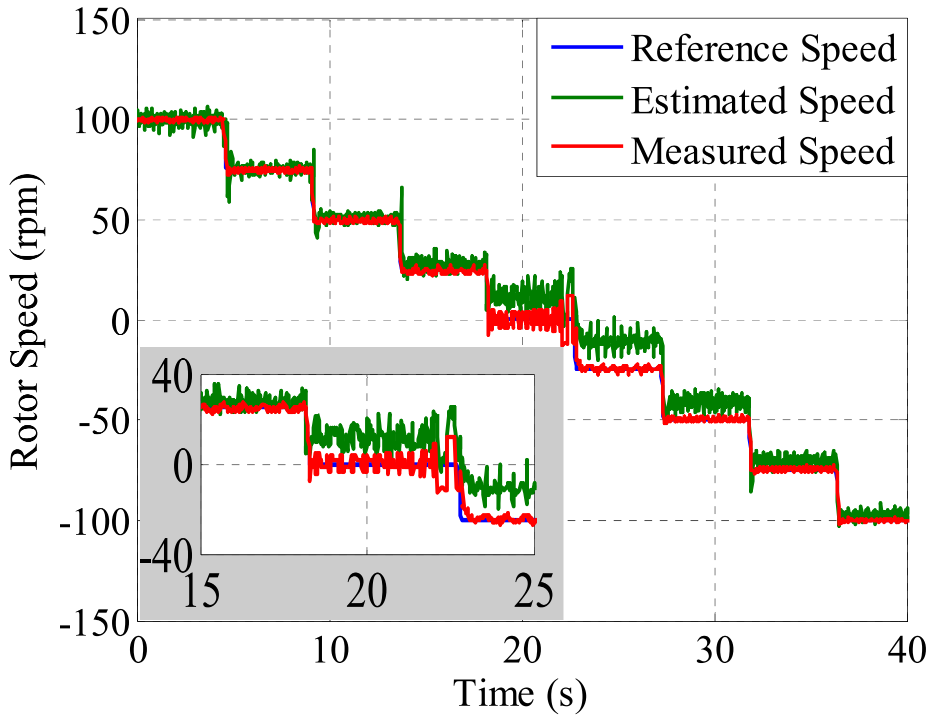


(a)

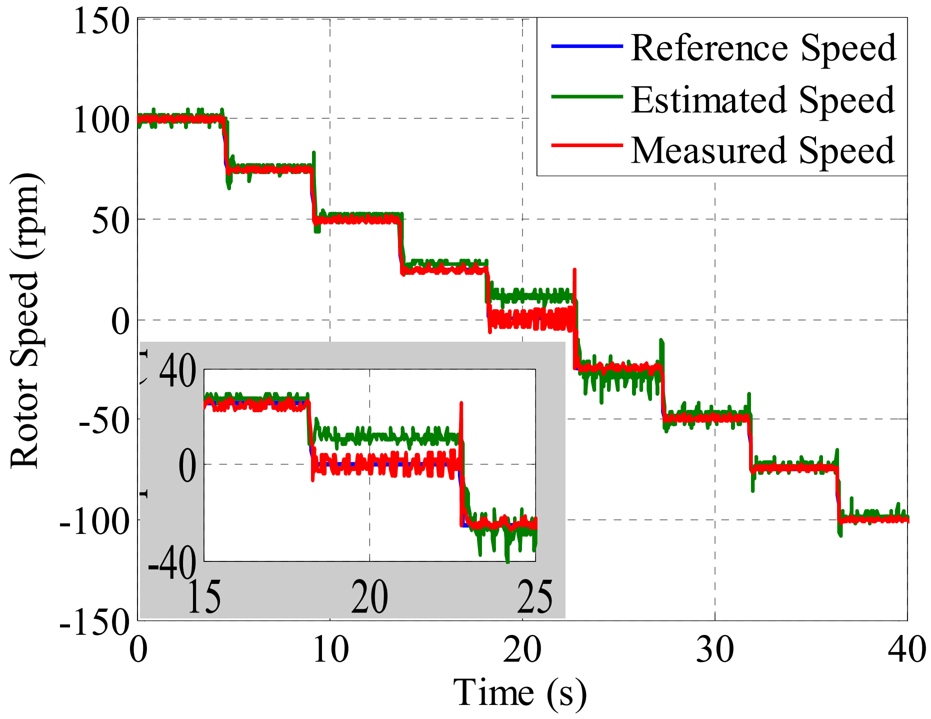


(b)

Fig. 7.3 Open loop estimation, 20 r/min and 75% load, estimated speed frequency spectrum (a) Classical MRAS (b) MP-MRAS

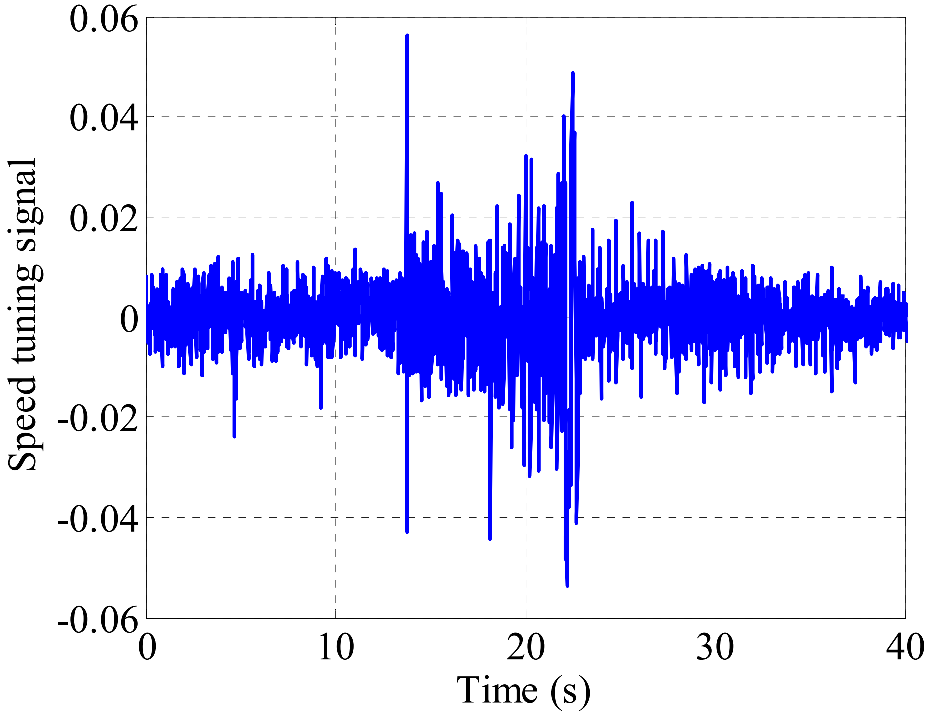


(a)

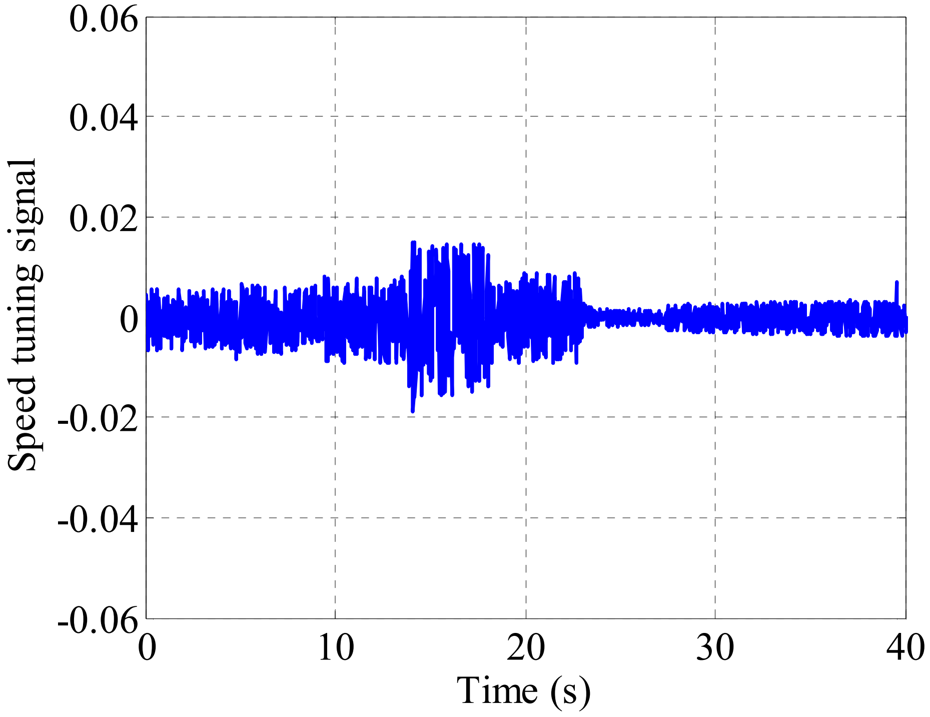


(b)

Fig. 7.4 Open loop estimation 63% load, low speed motoring and regenerating operation, rotor speed (a) Classical MRAS (b) MP-MRAS

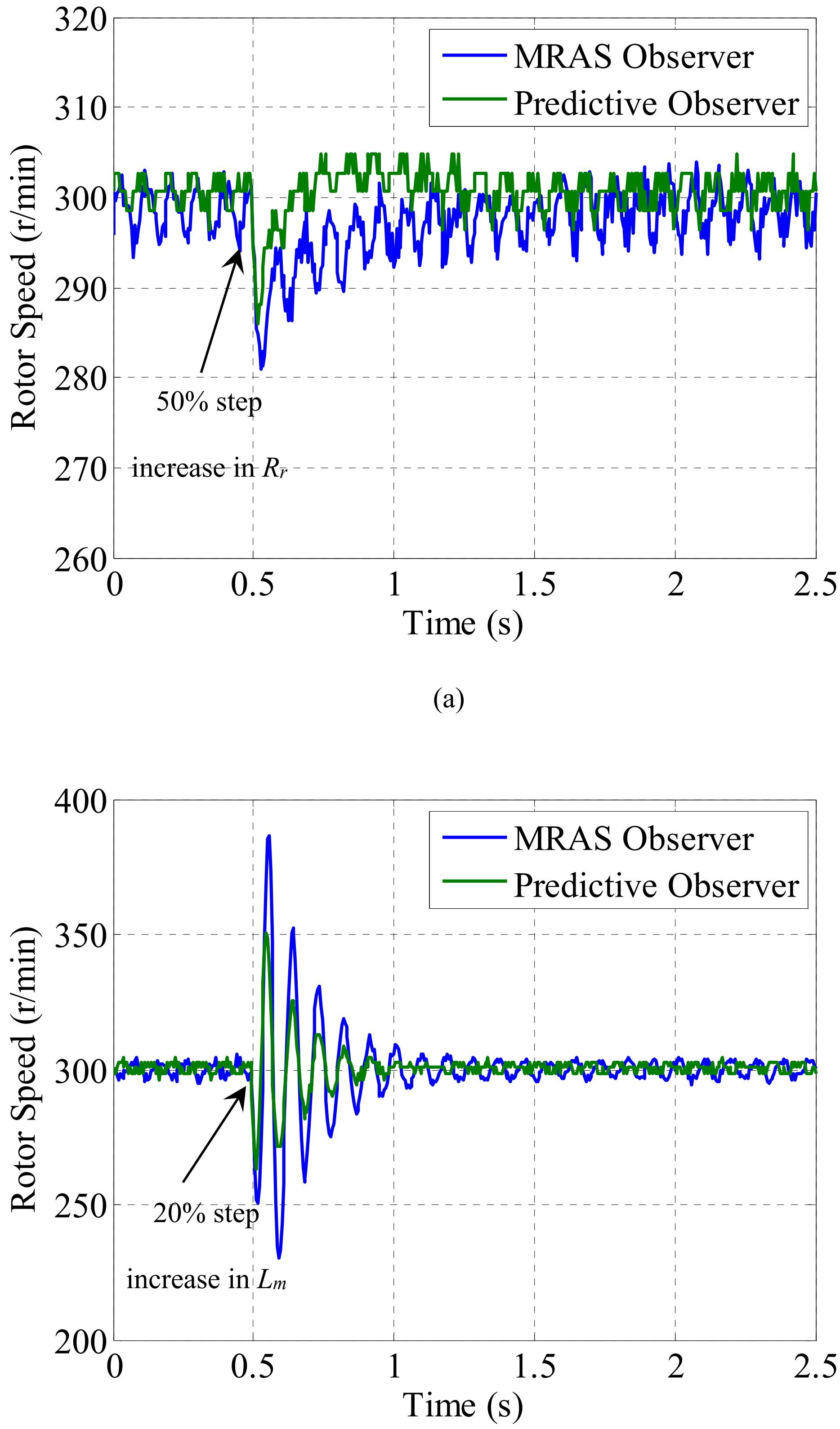


(a)



(b)

Fig. 7.5 Open loop estimation, 63% load, low speed motoring and regenerating operation, speed tuning signal (a) Classical MRAS (b) MP-MRAS



(a)

(b)

Fig. 7.6 Open loop estimation, 300 r/min and full load, effect of motor parameters variation. (a) 50% Change in R_r. (b) 20% Change in L_m

7.2.2 Sensorless observer operation

In this operation mode, the FOC scheme is driven by the estimated speed. The performance of both observers is tested at different speeds and load conditions.

Figs. 7.7-7.9, show the sensorless operation of both schemes when the drive is subjected to 75% of the rated load at 5% of the rated speed (75 rpm). Once again the MP-MRAS observer shows a better performance by reducing the oscillation in the estimated speed before and after applying the load, and this improvement appears more clearly in the frequency spectrum of Fig. 7.9. From Fig. 7.8 it can be also noticed that the predictive observer can produce better tracking between the adaptive and the reference model by keeping the speed tuning signal as small as 0.009 at all the operation points whereas it reaches to 0.08 in the classical observer when the load is applied.

In Figs. 7.10-7.11, the speed reference is changed from 2.6% (40rpm) to 6.6% (100rpm) of the rated speed, when rated torque is applied. At both speeds the new observer shows a better performance when comparing the oscillations in the estimated speeds and speed tuning signals.

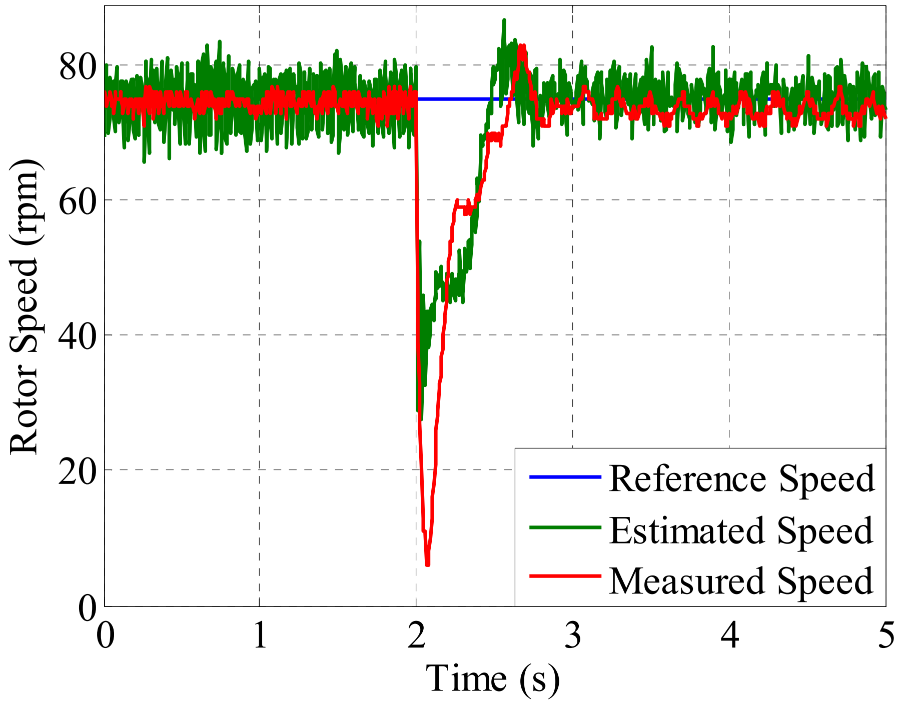
In comparison with sliding mode-MRAS observer results presented in [42], this observer offers much higher quality speed estimation with a reduced level of noise. Furthermore, a low pass filter is not needed to smooth out the estimated rotor speed. The proposed scheme is also less computationally demanding when compared to a rule-based fuzzy logic based-scheme [42].

To further validate the robustness of the proposed scheme against motor parameter variations, additional experimental tests have been carried when the machine is running at 30 rpm and no-load in sensorless mode of operation. In the first test, Fig. 7.12, a 50% step change has been applied to the rotor resistance R_r in the observer model. It is evident from Fig. 7.12 that the MP-MRAS observer is far less affected by the rotor resistance change, while the drive system loses stability in the case of the classical MRAS for the same level of R_r change. In the second test, Fig. 7.13, a step change of 50% has been applied to the stator resistance R_s in the observer model. It can be observed that the MP-MRAS scheme shows better performance with less oscillation in both the estimated and measured speeds. In the third test, Fig. 7.14, a step change of 20% has been applied to the mutual inductance L_m in the observer model. Fig.

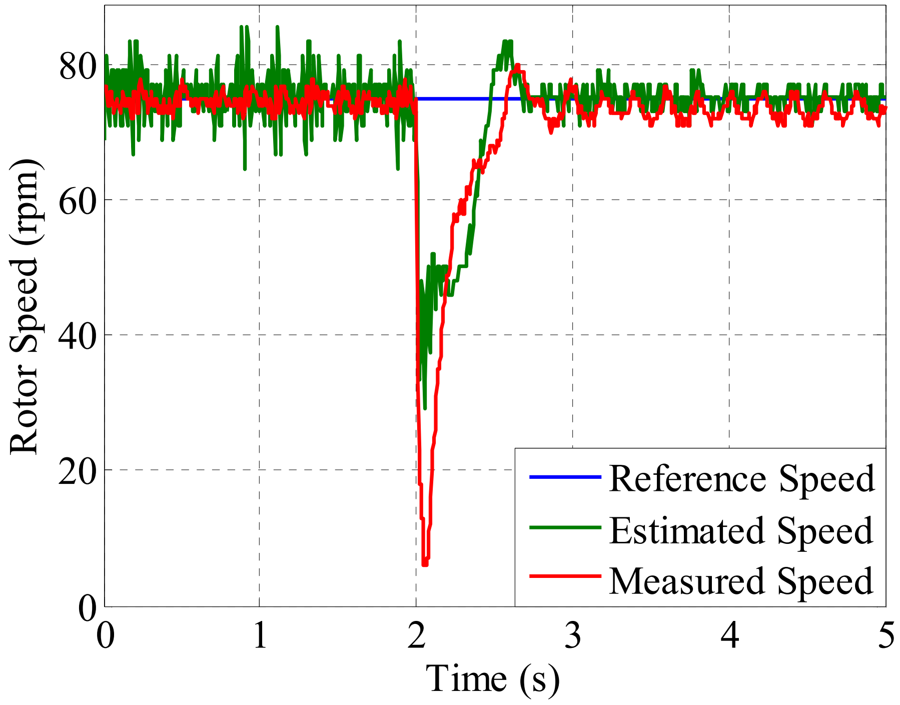
7.14 shows that the classical MRAS scheme has completely lost its stability after applying the change while the proposed MP-MRAS has exhibits much better performance.

To determine the minimum operating speed of the MP-MRAS observer, the reference speed is gradually reduced until the motor loses satisfactory operation. It is found that the minimum speed that can be achieved in the case of the MP-MRAS is 8 rpm compared to 25 rpm for the classical MRAS, a 68% improvement in low speed capability. Fig. 7.15 shows the sensorless operation of the proposed scheme at its minimum speed at both no-load and full-load conditions.

The effect of using the predictive observer on the speed controller bandwidth has been also tested. As the estimated speed of the proposed scheme is less noisy than the classical MRAS, this allows a further increase the PI gains of the speed control loop which will in-turn increase the maximum bandwidth that can be achieved. Experimentally, it has been found that the maximum bandwidth of the MP-MRAS observer is 156.68 rad/s compared to 85.63 rad/s for the classical MRAS.

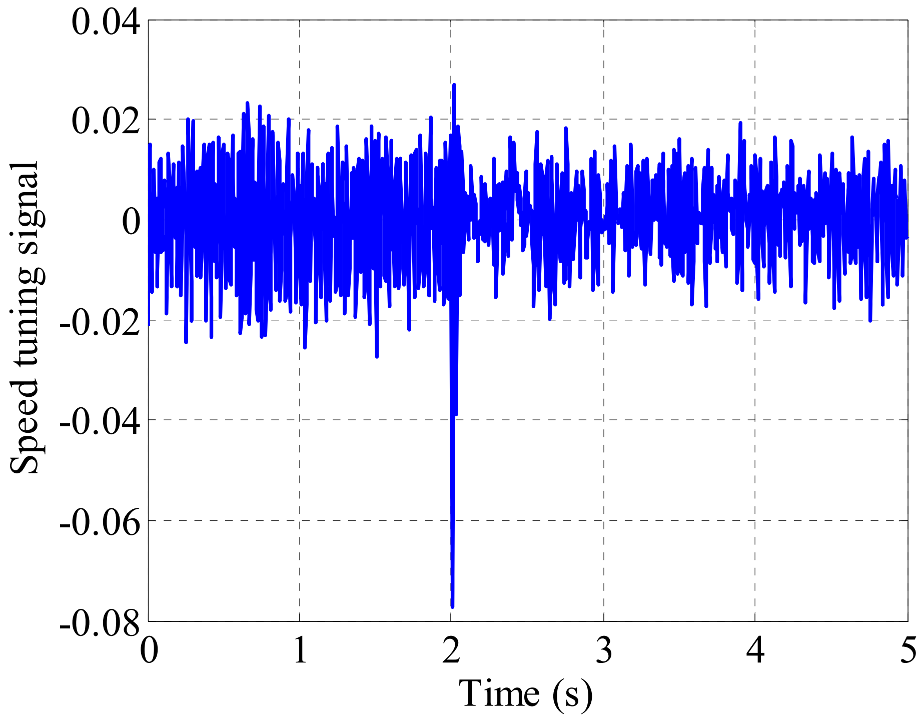


(a)

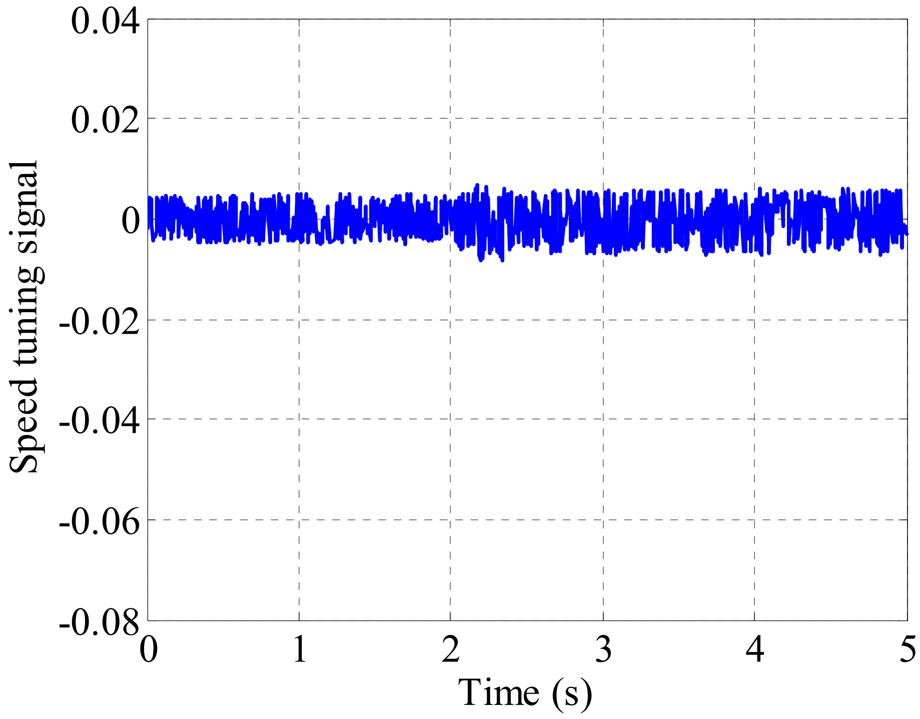


(b)

Fig. 7.7 Sensorless performance, 75 r/min and 75% Load, rotor speed (a) Classical MRAS (b) MP-MRAS

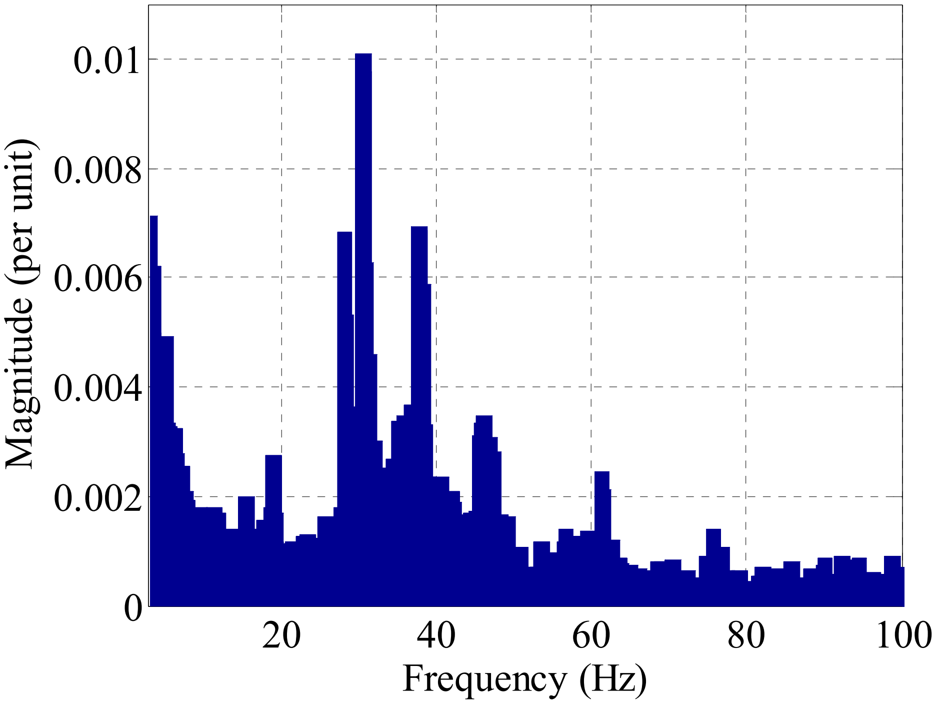


(a)

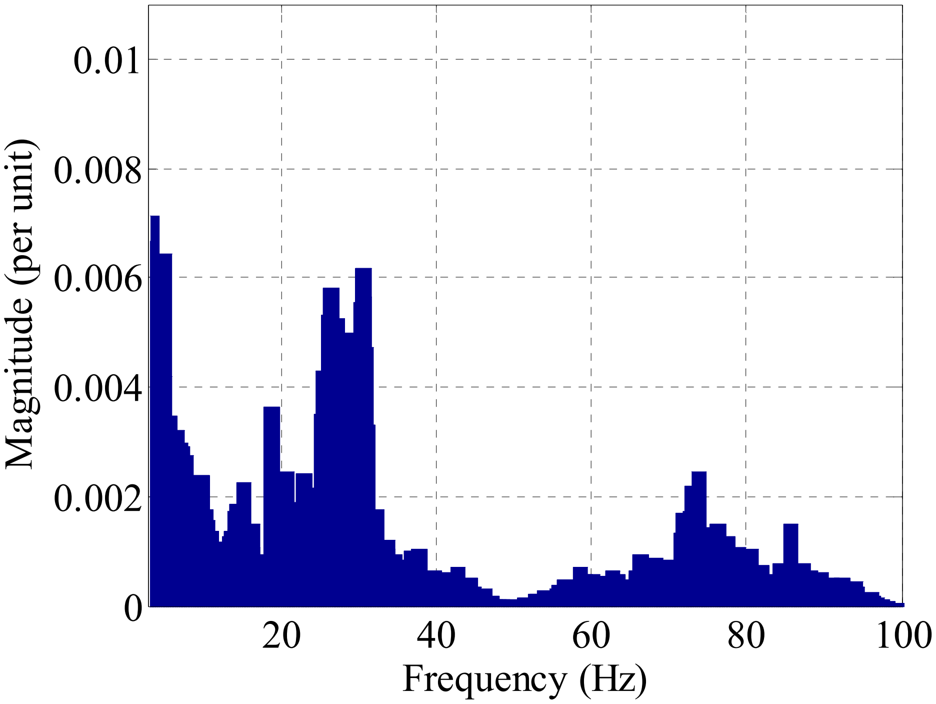


(b)

Fig. 7.8 Sensorless performance, 75 r/min and 75% Load, speed tuning signal. (a) Classical MRAS (b) MP-MRAS



(a)



(b)

Fig. 7.9 Sensorless performance,75 r/min and 75% Load, estimated speed frequency spectrum. (a) Classical MRAS (b) MP-MRAS

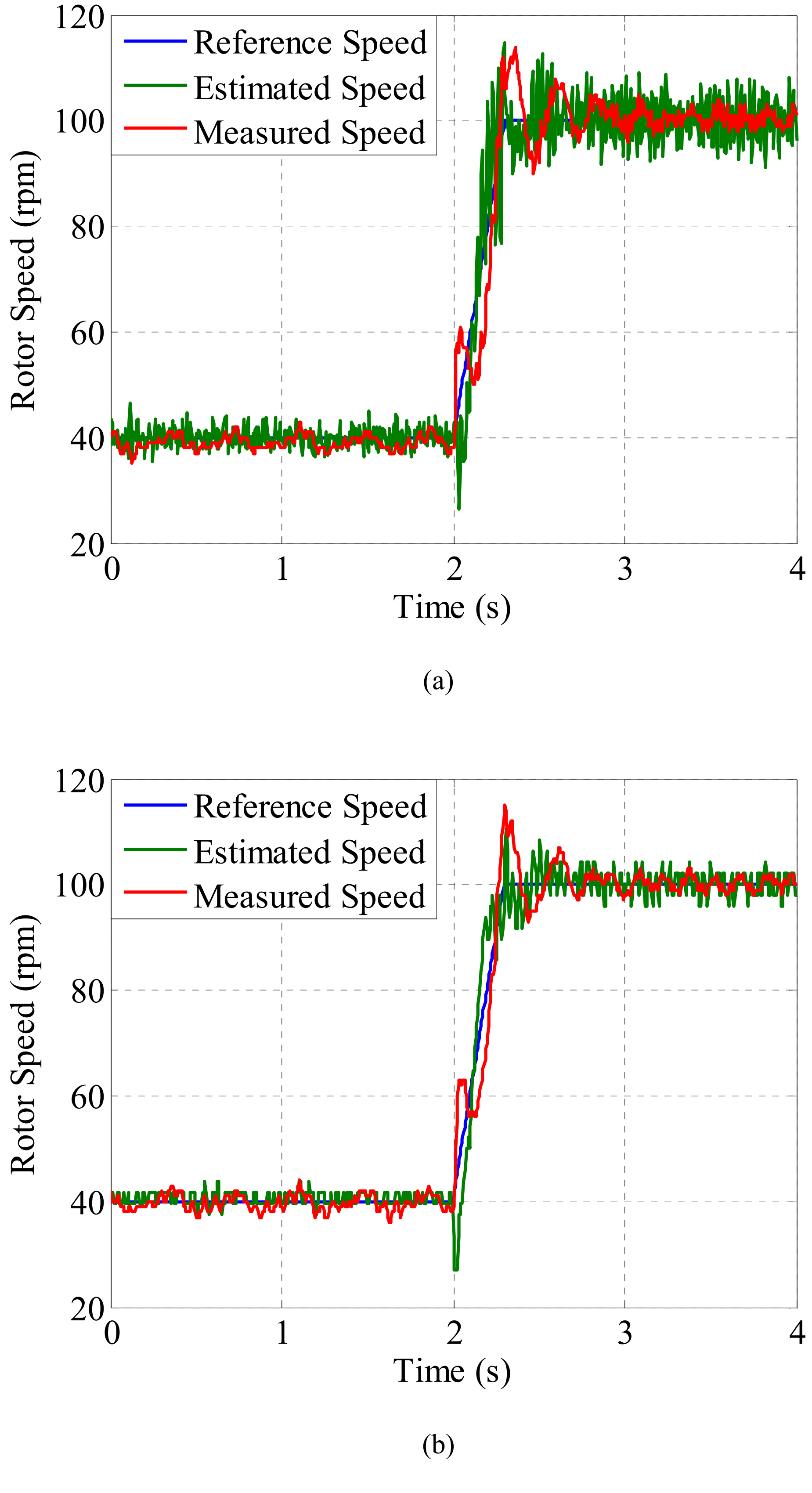
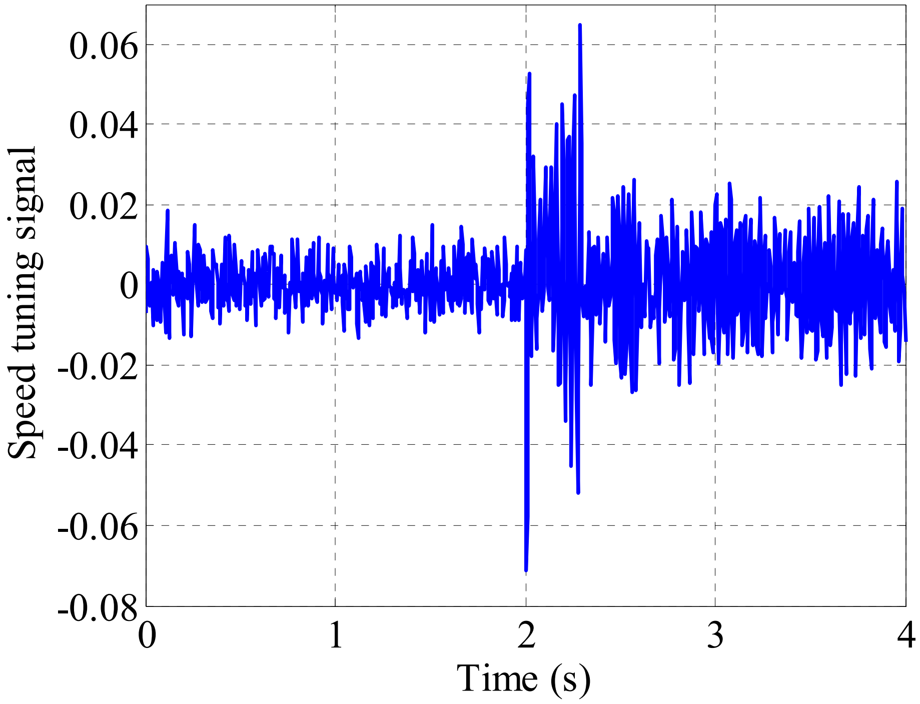
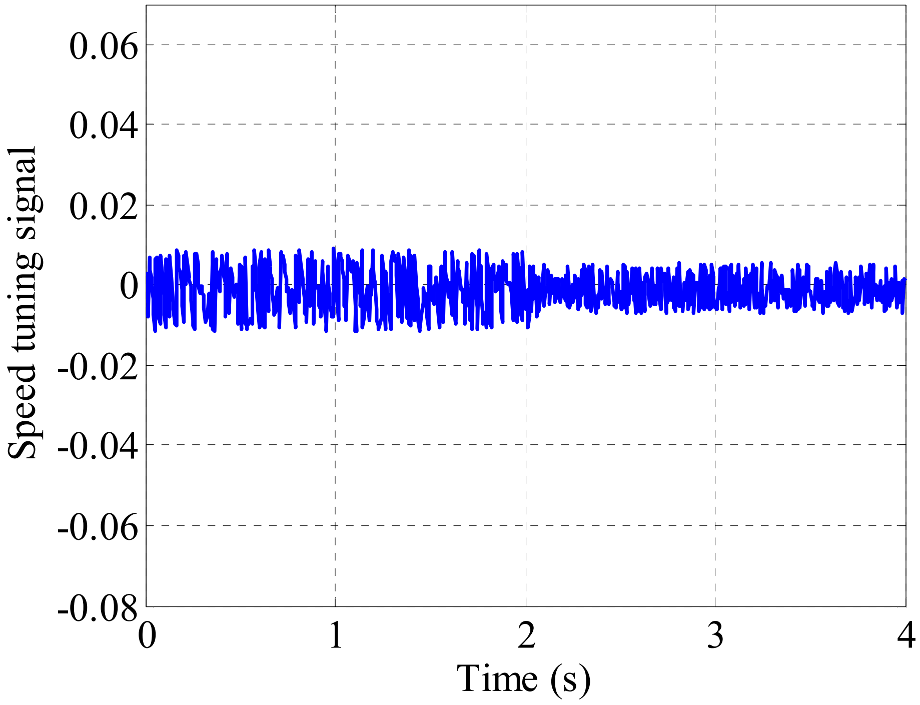


Fig. 7.10 Sensorless performance, reference speed change from 40 to 100 rpm at full load, rotor speed. (a) Classical MRAS. (b) MP-MRAS

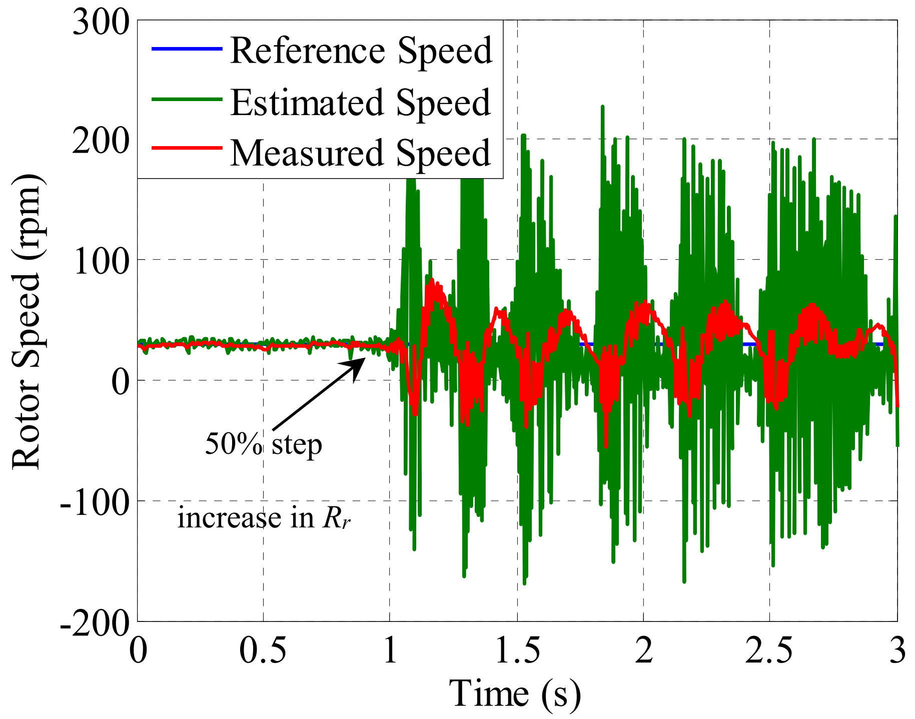


(a)

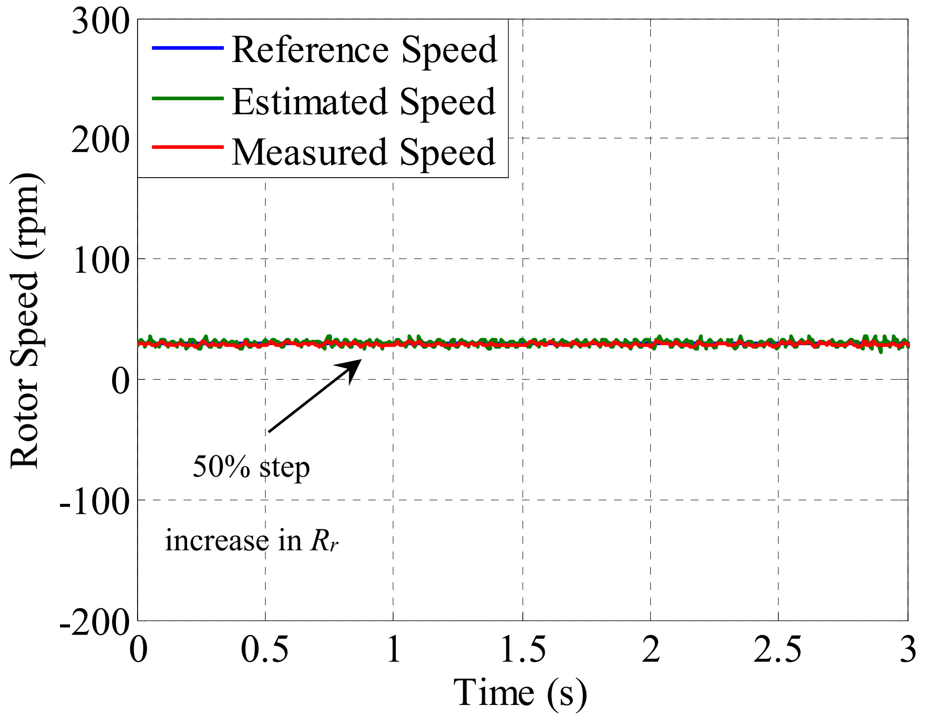


(b)

Fig. 7.11 Sensorless performance, reference speed change from 40 to 100 rpm at full load, speed tuning signal. (a) Classical MRAS. (b) MP-MRAS

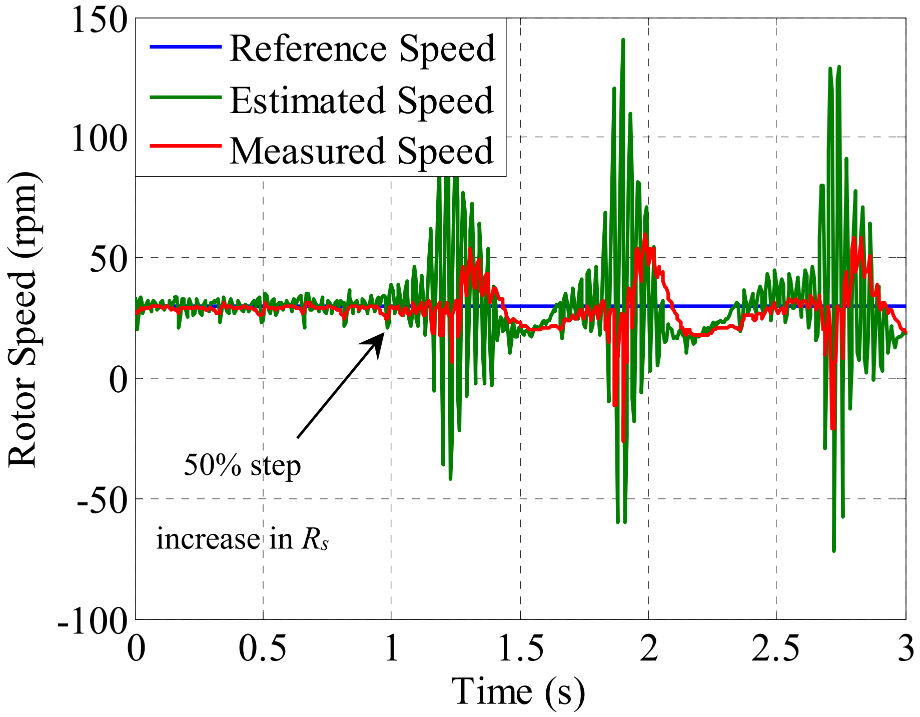


(a)

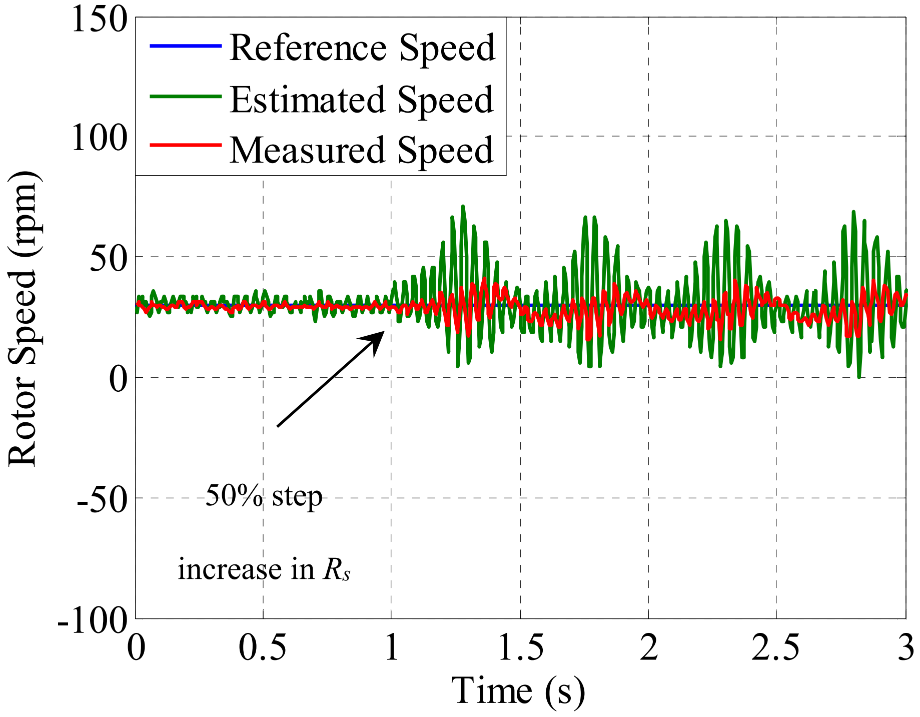


(b)

Fig. 7.12 Sensorless performance, the effect of rotor resistance change (a) Classical MRAS (b) MP-MRAS

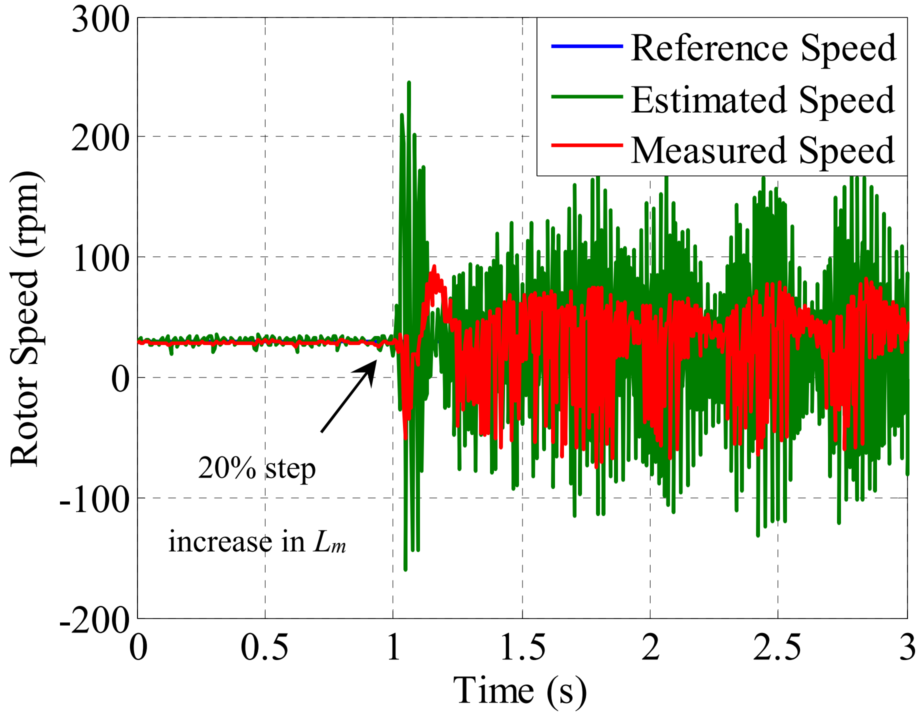


(a)

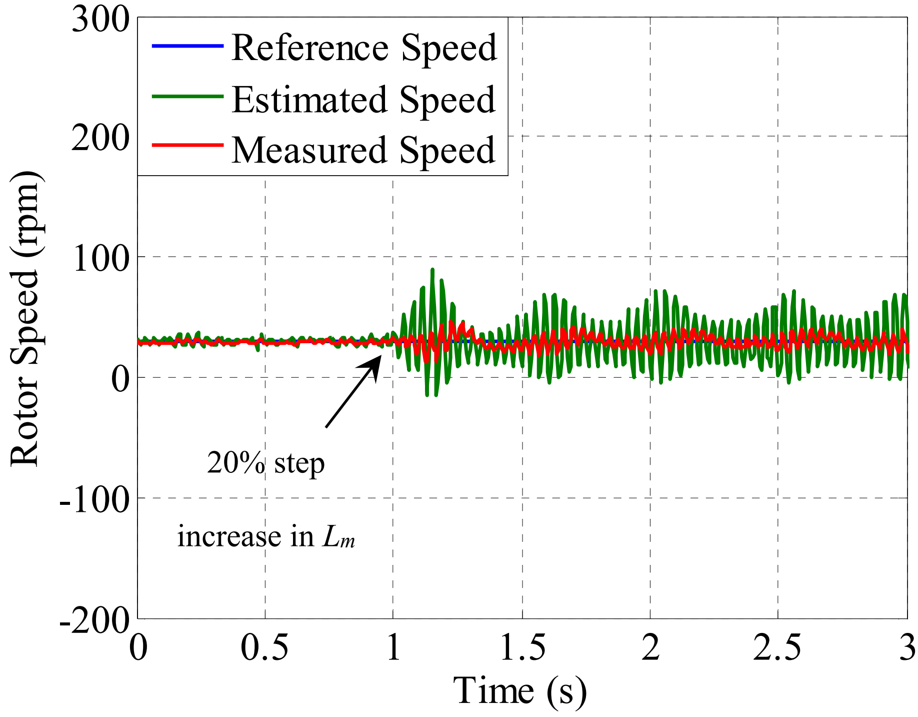


(b)

Fig. 7.13 Sensorless performance, the effect of stator resistance change (a) Classical MRAS (b) MP-MRAS



(a)



(b)

Fig. 7.14 Sensorless performance, the effect of mutual inductance change (a) Classical MRAS (b) MP-MRAS

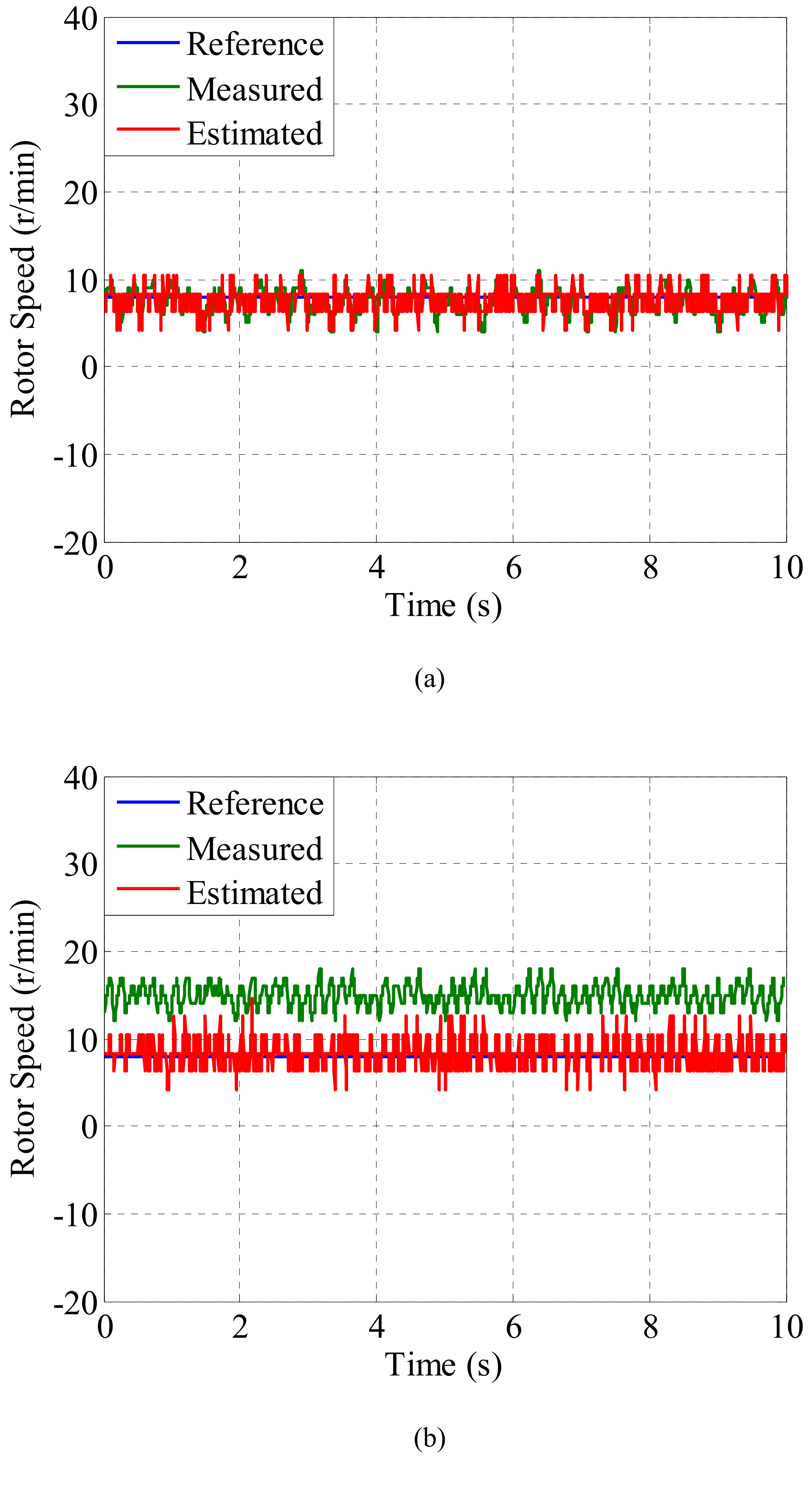


Fig. 7.15 Sensorless performance at minimum stable speed, MP-MRAS observer, (a) No-load (b) Full-load

7.3 Performance Comparison of the Normal and Improved MP-MRAS observers

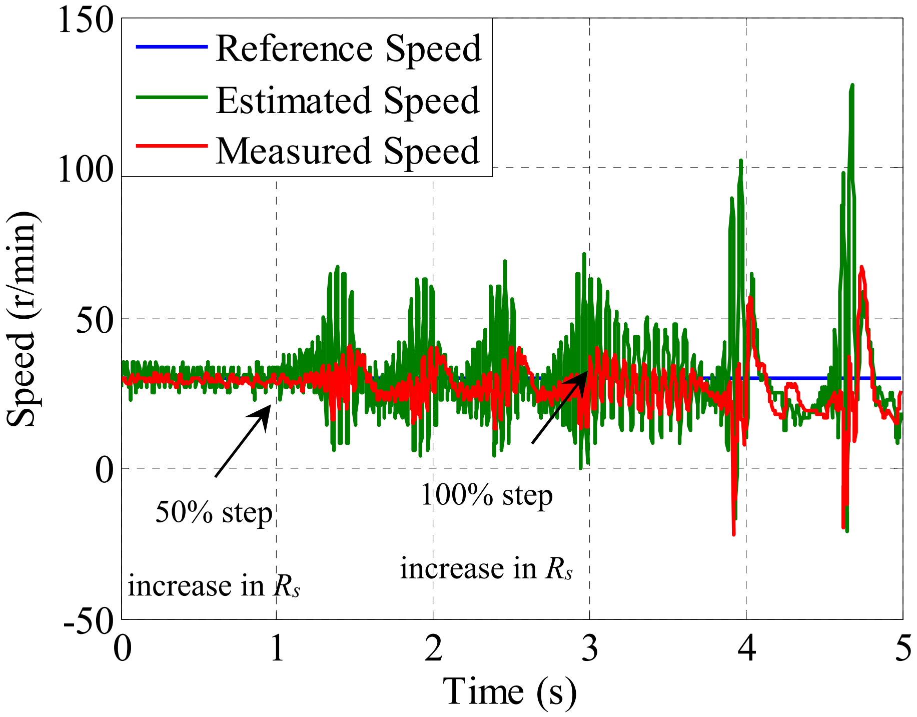
The parameters in the improved MP-MRAS observer are set to: K_P=600, K_i=120, τ_{low}=0.01 s, τ_{high}=0.058 s and the speed threshold 4 *r/min*. The tests are carried out in two stages. In the first stage, the robustness of the drive system against the motor parameters variation is tested, whereas in the second stage, the oscillations in the speed signal at different loading conditions are examined.

7.3.1 The robustness tests

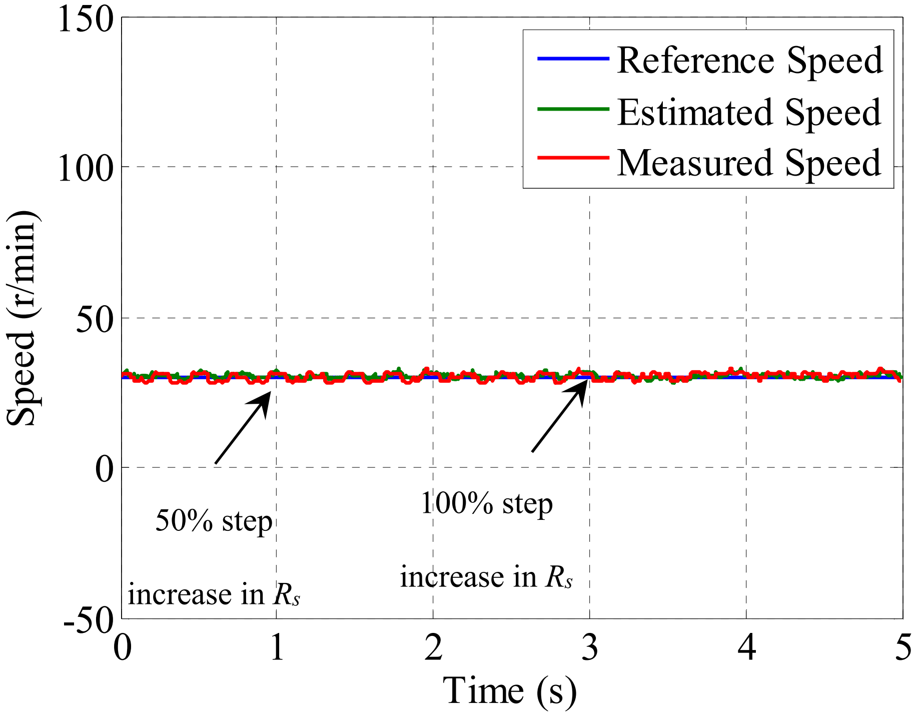
During the first test, while the drive is running at 30 rpm, 50% and 100% step changes have been applied to the stator resistance R_s. Figs. 7.16-7.17, show the speed and flux responses during the previous test where the normal MP-MRAS is applied in (a) and the improved MP-MRAS is applied in (b).

Fig. 7.16 shows that in the case of the normal MP-MRAS, as the rotor resistance increases the oscillation in the rotor speed also increases, and so does the distortion in the flux signals (Fig. 7.17), while in the case of the improved MP-MRAS, the speed and the fluxes are not affected by the resistance change.

In Figs. 7.18-7.19, the same test is repeated, but this time, to examine the mutual inductance variation effect on the speed and flux estimation quality. The mutual inductance is changed by 20% and 50% and the speed and the fluxes signals are recorded. From Fig. 7.18 it seems that the magnitude of the oscillation in the speed increases as the mutual inductance increase in the case of the normal MP-MRAS observer, whereas the speed is not affected by the inductance changes when the improved observer is used. The same applies to the estimated rotor flux, Fig. 7.19, where the fluxes are massively distorted when the inductance is changed in the normal MP-MRAS case and they maintain their sinusoidal form when the improved observer is applied.

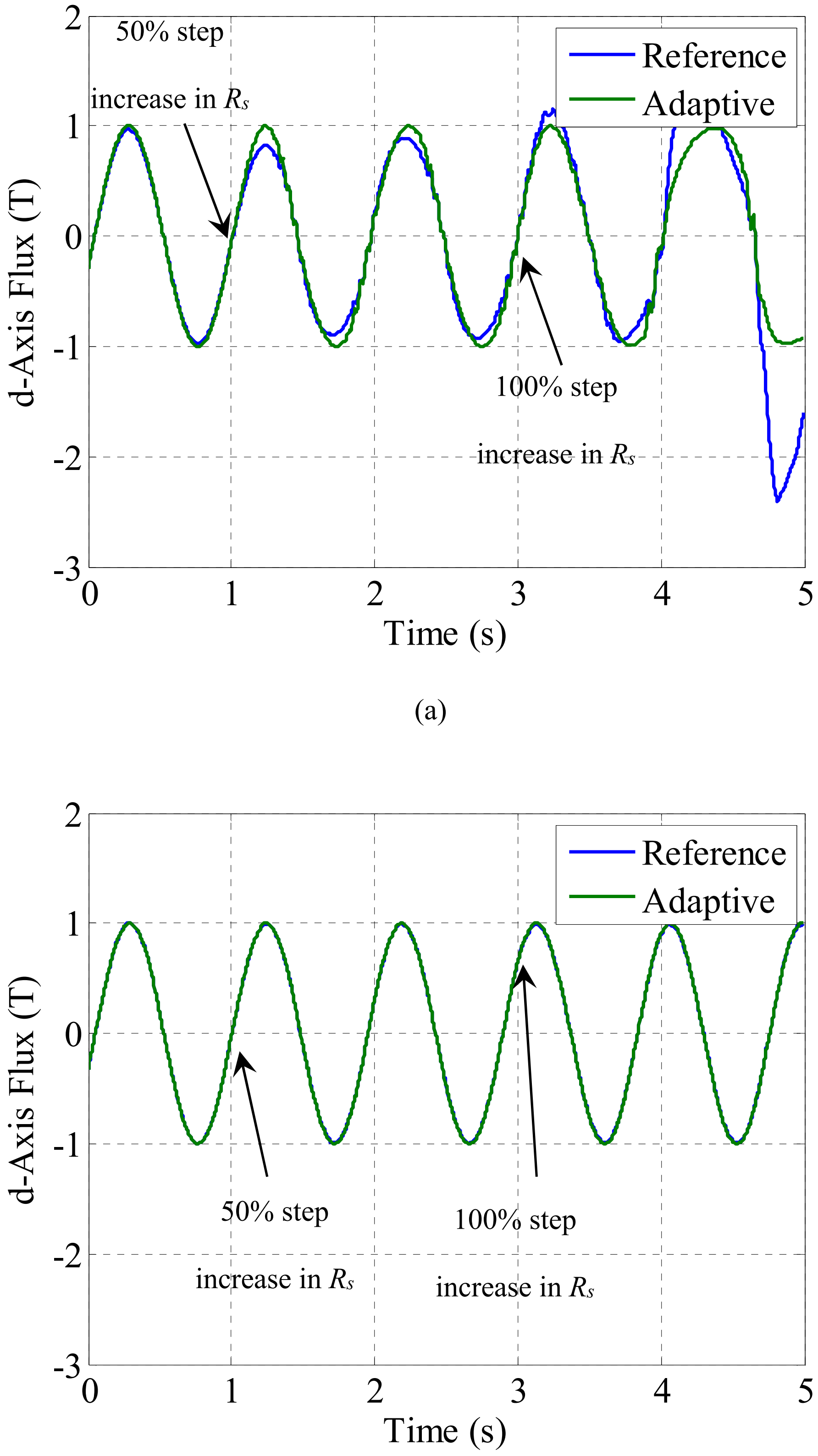


(a)



(b)

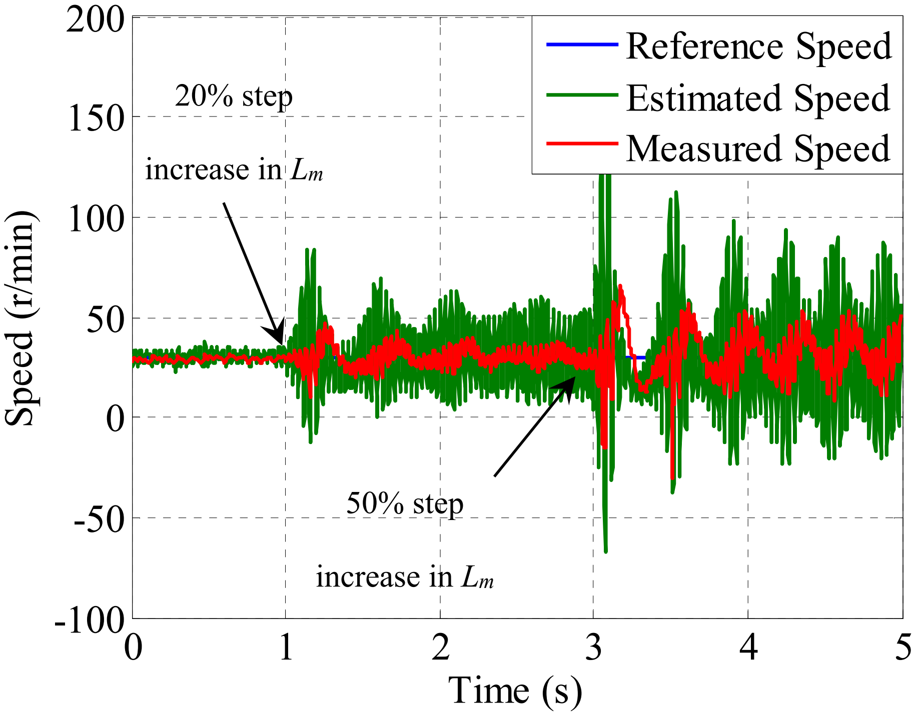
Fig. 7.16 Sensorless operation, rotor speed, R_s variation, 30 rpm and no-load operation. (a) normal MP-MRAS. (b) improved MP-MRAS



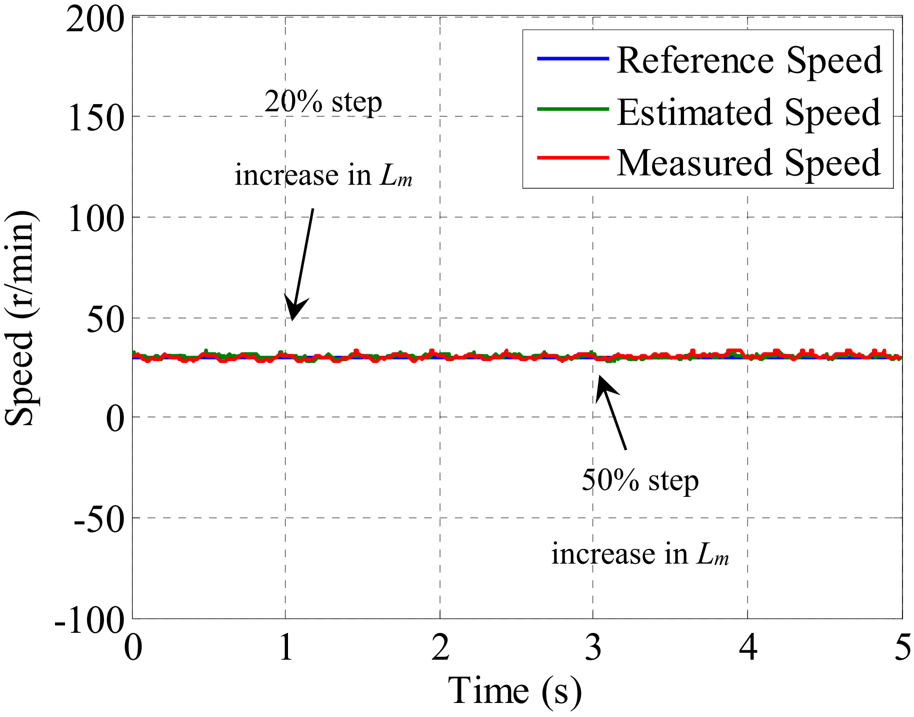
(a)

(b)

Fig. 7.17 Sensorless operation, d-axes rotor flux, R_s variations, 30 rpm and no-load operation (a) normal MP-MRAS (b) improved MP-MRAS

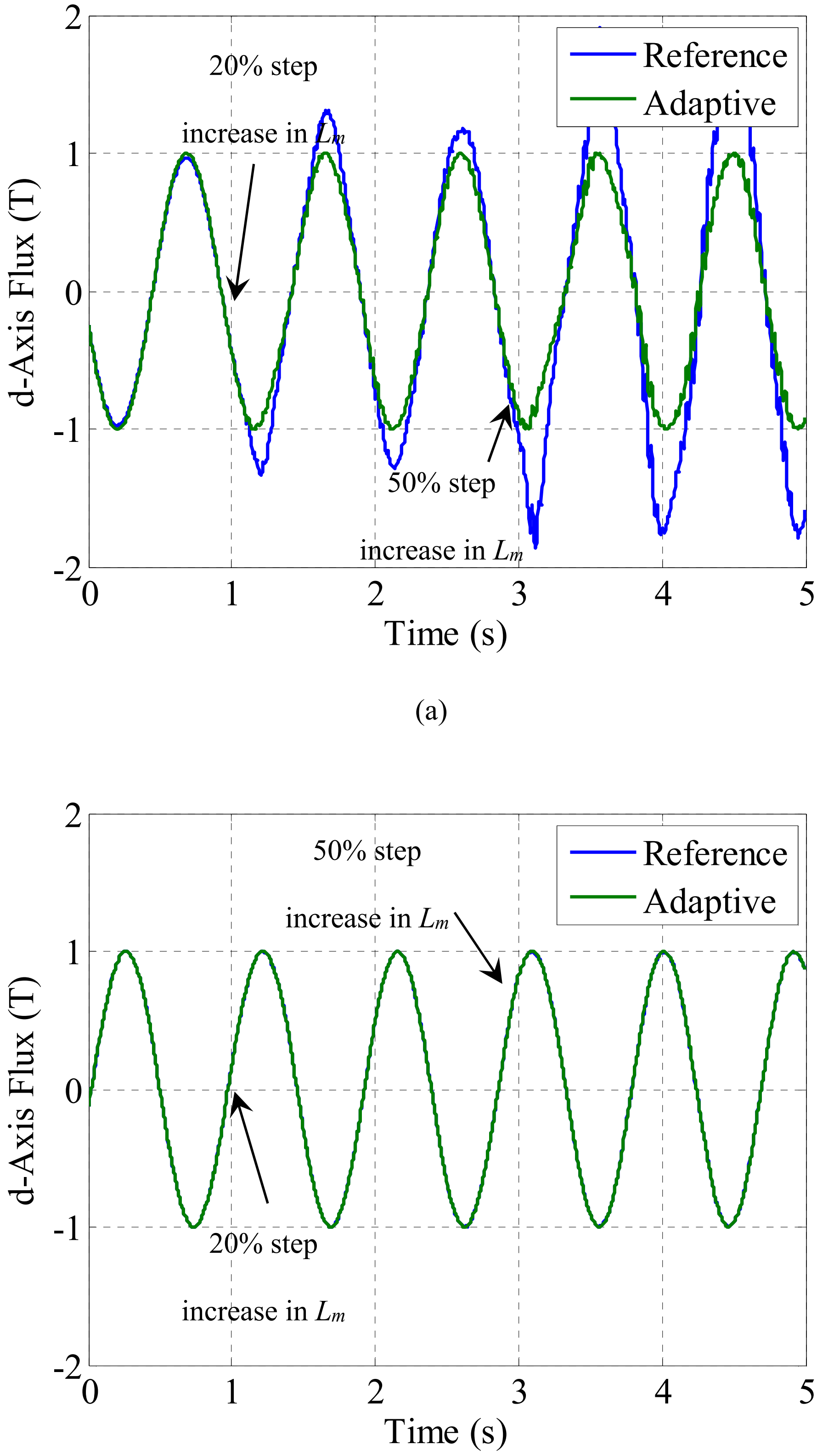


(a)



(b)

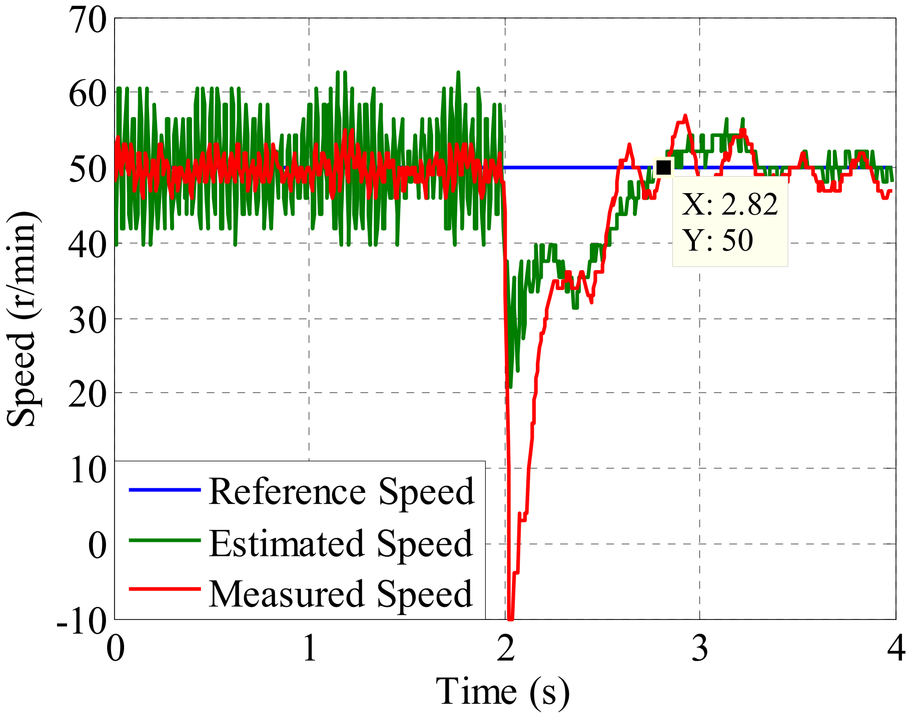
Fig. 7.18 Sensorless operation, rotor speed signal, L_m variations, 30 rpm and no-load operation (a) MP-MRAS (b) improved MP-MRAS



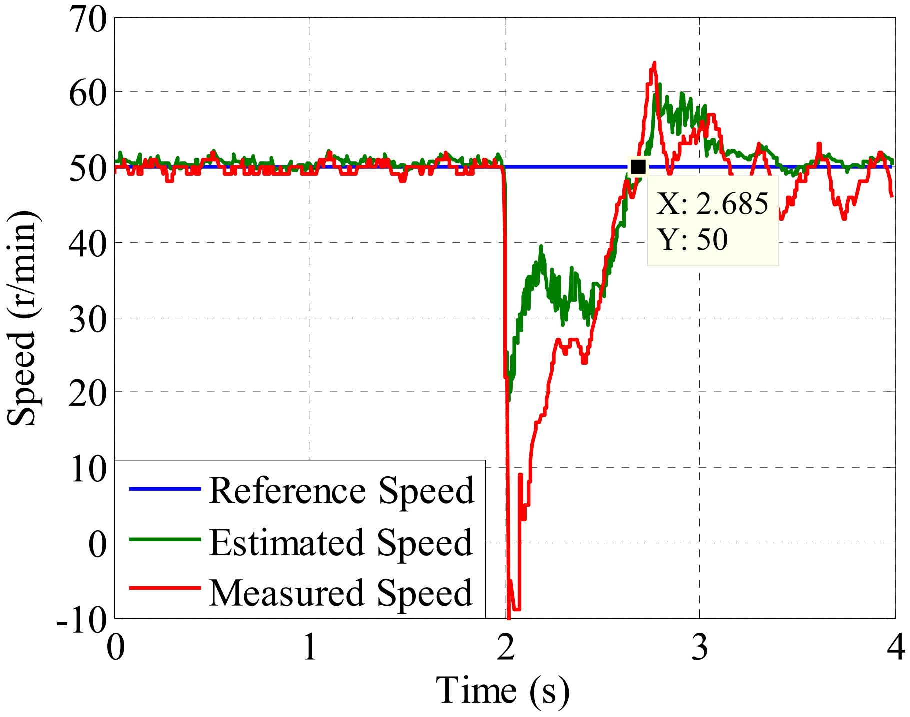
(a)

(b)

Fig. 7.19 Sensorless operation, d-axes rotor flux, L_m variations, 30 rpm and no-load operation (a) normal MP-MRAS (b) improved MP-MRAS



(a)



(b)

Fig. 7.20 Sensorless operation, rotor speed signal, 50 rpm and 75% step change in load (a) normal MP-MRAS (b) improved MP-MRAS

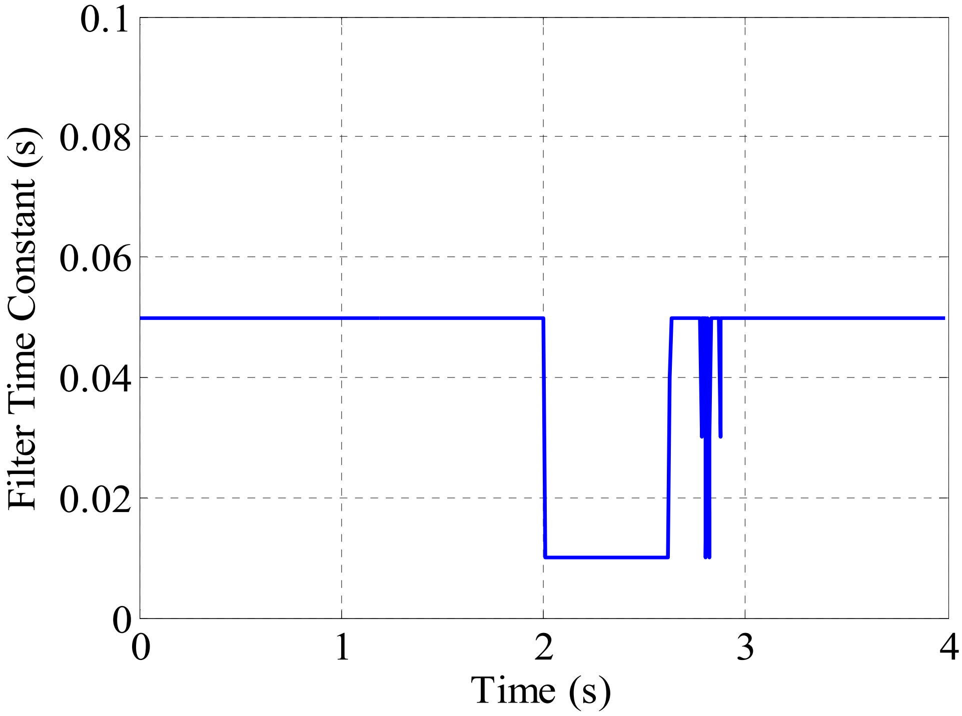


Fig. 7.21 The adaptive filter time constant, 50 rpm and 75% step increase in load

7.3.2 Testing the speed oscillations at different loading conditions

In the first test, while the machine is running at 30 r/min and no-load, 75% of the rated torque is suddenly applied. Fig. 7.20 shows the speed response of the drive for both observers, and it can be noticed that the MP-MRAS-based system shows a significant level of oscillations in the estimated speed especially before the load is applied compared to the improved MP-MRAS observer.

To further examine the quality of the estimated speed, the integral absolute error (IAE) is calculated over the 4 second time period of operation for both observers. The IAE in the case of the MP-MRAS observer is equal to 3789.8 r/min, whereas it is 2091.0 in the case of the improved MP-MRAS observer, which corresponds to a 44.8% reduction when the improved observer is applied.

The way the adaptive filter responds to the previous load disturbance is shown in Fig. 7.21 which represents the filter time constant τ versus time. From the figure it can be noticed that the filter applies the high time constant τ_{high} during the steady state operation which effectively reduces the oscillation in the estimated speed. After the load is applied the filter reduces its time constant instantaneously to make it equal to τ_{low} and this results in a faster transient response.

In the next test, the system response to a step change in the speed demand is examined for both observers. The speed demand is set initially to 30r/min and at 2s it is suddenly changed to 60 r/min. Fig. 7.22 shows the system response for both observers, and once more the system with the improved observer shows less oscillation in the estimated speed compared to the normal MP-MRAS observer. The IAE of the estimated speed for the normal MP-MRAS observer is 2318.0 r/min, compared to 738.4 in the case of the improved observer, which means a 68.1% reduction.

The same test is repeated at full load for different speeds in Fig. 7.23. The speed demand is set at first to *40r/min* and at *2s* it is suddenly changed to *80 r/min.* The figure shows that the amount of the oscillation in the estimated speed is less in the case of the improved observer compared to the normal MP-MRAS observer. The IAE of the estimated speed for the normal MP-MRAS is 1705.7 *r/min*, whereas it is equal to 1335.9 *r/min* for the improved observer.

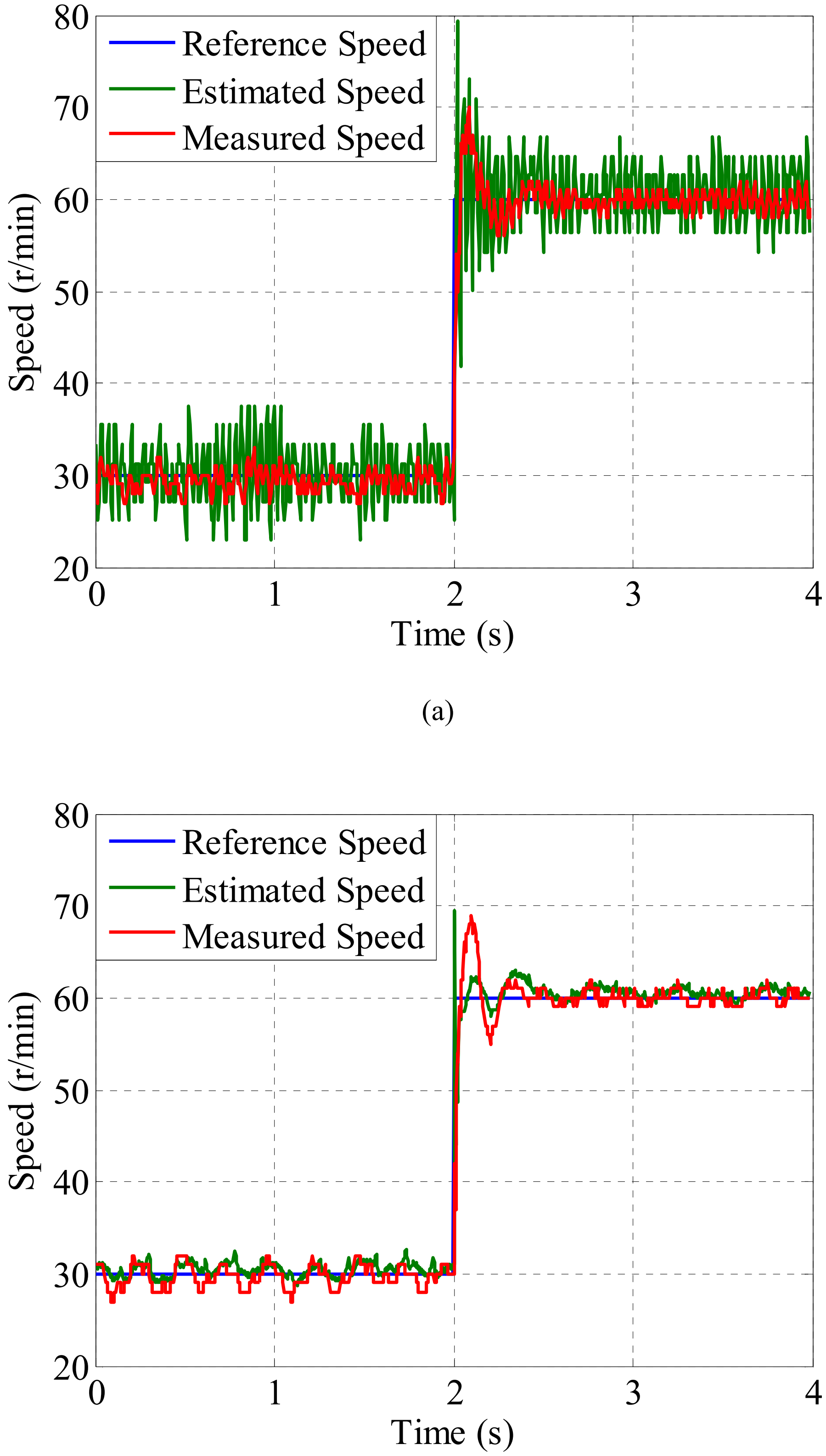
Since there is a link between the controller maximum bandwidth and the amount of the oscillation in the estimated speed, as it was reported in [97], the bandwidth of the speed control loop is measured for both observers. It was found that the maximum bandwidth of the system when the improved MP-MRAS observer is applied is 223.626 rad/s, whereas it is equal to 156.68 rad/s for the normal observer. This means a 42.73% further improvement in the controller bandwidth can be achieved if the improved MP-MRAS observer is applied.

7.4 Performance Comparison of the Simulation and practical results

In general, both the simulation and the practical results show the same outcomes when the performance of proposed observer is compared with the classical one. These outcomes can be summarized as reduction in the oscillation in the estimated and real speed signals and keeping the speed tuning signal at the minimum level even during the transient events.

However, comparing Figs 5.5 and 7.4 and Figs 5.3 and 7.1 shows a steady state error in the estimated speed that appears only in the practical results at very low speed and not in the simulation results. This steady state error is explained by the low pass filter used in the reference model which exists only in the practical circuit. This filter is needed to solve the problem of the pure integration which does not present in the simulation.

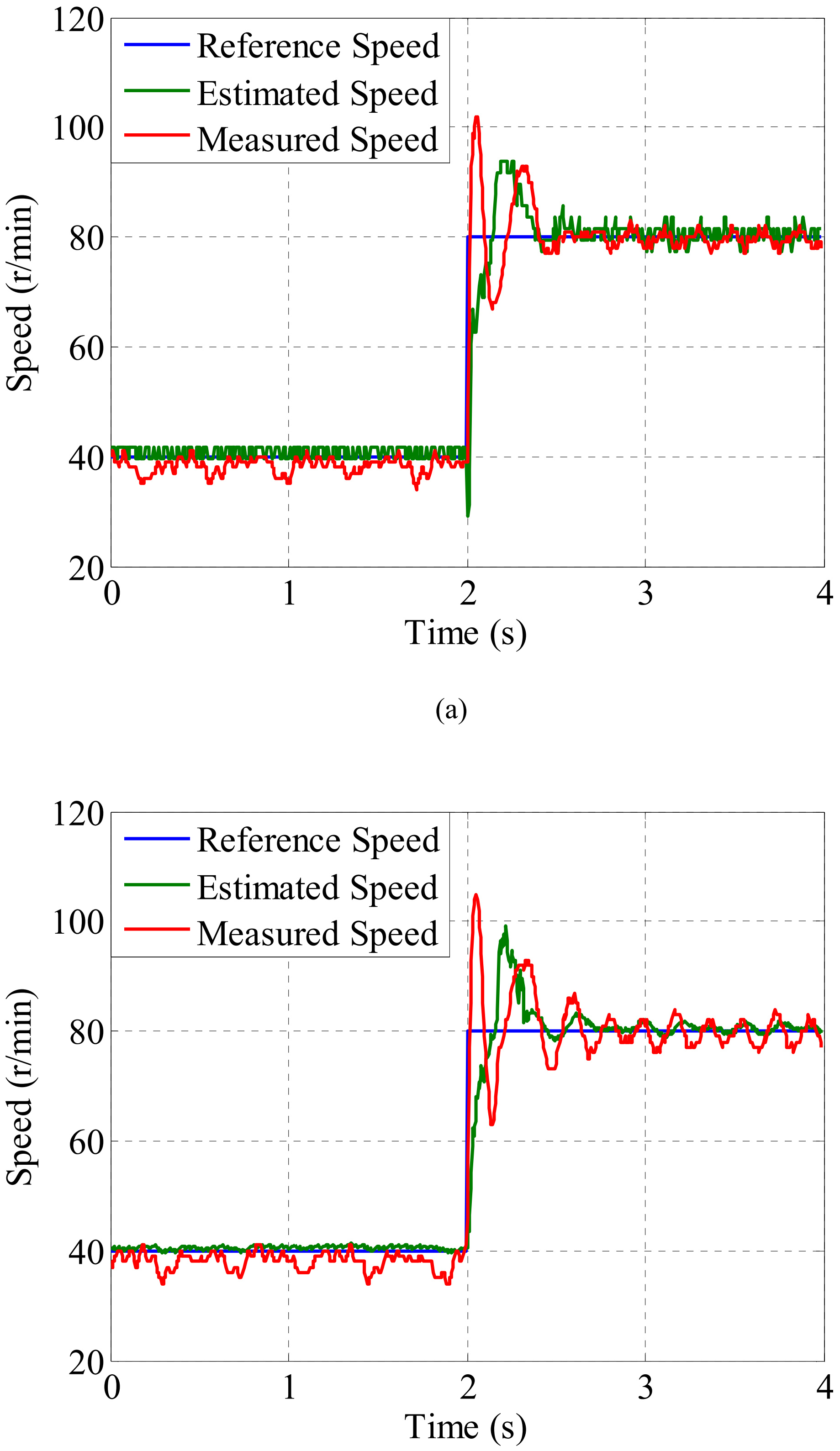
It can be also noticed from all the figures that, in the practical results the level of the oscillation is higher in both the estimated and real speeds in compare with the simulation results. These oscillations can be explained by two different factors. The first one is that the current, voltage and speed sensors were considered ideal in the simulation while in the real circuits these sensors can pick up large amount of high and medium frequency noise. In addition, the wrong calibration of these sensors can cause additional noise especially in the frequencies close to the fundamental. The second factor is the inverter which was considered ideal in the simulation, while in the real inverter, the switching events cause a considerable amount of high frequency noise. Also the real inverter causes additional noise due to the dead time and inverter nonlinearity effects. These effects were compensated in the practical circuit as mentioned in chapter 6. However, this compensation does not solve this problem completely.



(a)

(b)

Fig. 7.22 Sensorless operation, rotor speed signal, step change from 30 r/min to 60r/, no-load (a) normal MP-MRAS (b) improved MP-MRAS



(a)

(b)

Fig. 7.23. Sensorless operation, rotor speed signal, step change from 40 r/min to 80r/min, full load (a) normal MP-MRAS (b) improved MP-MRAS

7.5 Summary

In this chapter, detailed experimental tests were carried out to compare the performance of three MRAS-based observers; the classical rotor flux based MRAS, the MP-MRAS and the improved MP-MRAS observers. The experimental results section was divided into two parts or subsections. In the first part, a performance comparison was carried out between the classical MRAS and the MP-MRAS observers, whereas the performance of the improved MP-MRAS was compared with the normal MP-MRAS in the second part.

The results of the first comparison between the classical and MP-MRAS observers show a better estimation quality of the rotor speed with a significant reduction in steady-state oscillations without affecting the dynamic response as a minimum speed tuning signal is ensured in both transient and steady-state conditions. Hence, a higher maximum bandwidth of the speed control loop was achieved when the MP-MRAS observer was employed. Improved robustness against motor parameter variations was also noticed for the proposed scheme. This improvement is mainly due to the replacement of the PI controller in the adaptation mechanism by a search-based optimization algorithm. It is well reported in the literature that fixed-gain PI controllers are generally not robust to changes in system parameters [98]. However, the MP-MRAS showed oscillations in the estimated speed when it operates at light loading condition and it showed sensitivity to the motor parameters variation, to a certain extent, due to the use of the voltage equation in the reference model.

The results of the second part shows that compared to MP-MRAS observer, the improved MP-MRAS scheme is robust against motor parameter variation. Employing the proposed scheme provides better rotor speed estimation with a significant reduction in the oscillations at different loading conditions, especially light loads, without affecting the dynamic response. Also a higher maximum bandwidth has been achieved for the speed control loop when the improved predictive estimator is applied as a result of noise reduction. Therefore the proposed improved MP-MRAS scheme is suitable for the sensorless control of IM drives by providing robust and smooth speed estimation.

From above it can be concluded that, the new scheme improves the steady state performance of the drive by both reducing the oscillations in the estimated and real speed signals and reducing the effect of the motor parameters variation on speed estimation. In addition, the proposed scheme improves the transient response of the drive by increasing the maximum bandwidth of the speed control loop which means the controller will react faster during the transient events.

CHAPTER 8

Summary and Future Work

8.1 Introduction

In this thesis, research was conducted to investigate the application of a new model predictive MRAS (MP-MRAS) speed and position observer for sensorless IM drives. The main purpose of proposing the new observer is to improve the adaptation mechanism design by firstly eliminating the PI controller in the adaptation mechanism and secondly introducing a search-based optimization algorithm which ensures a minimum speed tuning error signal at each sampling period. The scheme was tested at different speeds and loading conditions and a focus was given to operation at low and zero speed regions. The aim of this chapter is to give a summary of the primary results, analysis and findings of the research and to give few recommendations for some possible future work.

8.2 Research Summary and Conclusions

In chapter one, it was mentioned that sensorless control was introduced to the IM drives to increase the system reliability and reduce its cost. Sensorless strategies were classified into model-based and spectral analysis based techniques. The focus was given to the model-based strategies in which the stator currents and voltages are employed along with the machine mathematical model to extract the rotor speed and position information. These strategies, however, usually fail at low speeds as the motor rotor model becomes unobservable. Predictive control was also presented in this chapter as a powerful control strategy that had been successfully employed to improve many of the problems associated with IM control.

In chapter two, MRAS observers were presented as simple and powerful algorithms that offer accurate rotor speed and position estimation without high computational effort. Three different types of MRAS observer were reviewed, which are the rotor flux-based MRAS, back EMF-based MRAS and reactive power-based MRAS. All of these MRAS observers suffer a number of problems at and around zero speed. These include pure integration, parameter sensitivity and inverter nonlinearity problems. The various solutions suggested to solve these problems were presented in this chapter. Also a survey of the different sensorless control systems that employ predictive controllers to improve the drive performance was provided.

Chapter three introduced the induction machine dynamic modelling using space vector theory. The machine model was then used to explain the principle of vector control. The two-axis machine model was also used to describe the rotor flux MRAS speed observer principle and to formulate the equations of the observer mathematical model. This observer is the most common strategy for sensorless control. Making use of Popov's hyperstability theory, a PI-based adaptation mechanism was derived to estimate the rotor speed. In the last part of this chapter, a discrete-time representation of the machine and the observer is provided.

In chapter four, a review of model predictive controllers was presented and a focus was given to finite control set model predictive controllers. Next, the proposed MP-MRAS speed and position observer was described. This observer is based on finite control set-model predictive control principle. The rotor position is calculated using a search-based optimization algorithm which ensures a minimum speed tuning error signal at each sampling period. This eliminates the PI controller employed in the classical MRAS observers’ adaption mechanism. Also in chapter four an adaptive filter was introduced to the design of the speed extraction mechanism to reduce rotor speed oscillations at light loading conditions. An adaptation mechanism was proposed to change the filter time constant depending on the dynamic state of the system. Furthermore, a voltage compensating method was employed in the reference model of observer to address the problems associated with sensitivity to motor parameter variation. This modified scheme was referred to as improved MP-MRAS observer.

In chapter five, a simulation using Matlab/Simulink was carried out to examine the performance of the proposed schemes compared to the classical rotor flux MRAS scheme. This study assumed an ideal inverter and ideal measurements. The scheme was tested in the open loop and closed loop conditions. The results showed that the proposed scheme produced a more accurate rotor speed estimation. The speed tuning error was very small even during the transient operation conditions. A significant reduction in steady-state oscillation was also achieved without affecting the dynamic response. Improved robustness to motor parameter variations was also demonstrated for the proposed scheme.

In chapter six, the experimental setup used in validating the proposed observer scheme was presented. The hardware was divided into three different parts, the motor rig which is based on a 2.2 *kW* squirrel cage IM, the motor drive unit which consists of the power converter and which is responsible for delivering the electrical power to the IM, and finally the control unit which is based on an eZdsp F28335 board.

In Chapter seven, different experimental tests were carried out to examine the proposed speed and position observer in a real-time application. These tests included different speed regions and loading conditions with particular interest given to low speed and regeneration operation. To avoid the problem of the pure integration associated with the flux calculation, a LPF was used with a very low cut-off frequency in the voltage model. In addition, the reference voltages from the controller were used in the voltage model to avoid the measurement of the real voltages. Compensation for the inverter nonlinearity and the dead-time was also considered. The gains of the PI controller in the classical MRAS observer were tuned to obtain the fastest possible dynamic performance. The results were obtained over two stages, in the first stage the performance of the normal MP-MRAS and the classical rotor flux MRAS observers were investigated, while the improved MP-MRAS observers was tested in the second stage. The results of the first stage showed better estimation quality of the rotor speed when the predictive observer was applied. This demonstrated a significant reduction in steady-state oscillations without affecting the dynamic performance. Hence, a higher maximum bandwidth of the speed control loop was achieved. The proposed observer also showed more robustness to motor parameters variation and more accurate speed estimation during regeneration. The second stage results showed that the improved MP-MRAS observer is both very robust to motor parameters variation and able to provide better rotor speed estimation. There is also a significant reduction in oscillation at different loading conditions, especially for light loads. A higher maximum bandwidth was achieved for the speed control loop compared to the normal MP-MRAS observer as a result of oscillation reduction. As a conclusion the proposed MP-MRAS observer is suitable for the sensorless control of IM drives by providing robust and smooth speed estimation.

8.3 Recommendations for Future Work

In this thesis a new predictive rotor flux MRAS speed observer was proposed for sensorless control of IM drives. This observer is based on finite control set-model predictive control principle. The rotor position is calculated using a search-based optimization algorithm which ensures a minimum speed tuning error signal at each sampling period. This search algorithm replaced the PI-based adaptation mechanism in the classical rotor flux-based MRAS. This work can be extended to include other types of MRAS observers that deploy a PI controller as an adaptation mechanism such as back EMF-based and reactive power-based MRAS observers. Furthermore, the principle of the proposed observer can be applied with different applications which require angle estimation such as in the grid-connected inverters, where the knowledge of the phase angle of the utility voltage is critical. The proposed predictive observer can provide an accurate and fast tracking of the utility angle information which is essential to ensure correct operation of the inverter controller. The work on this predictive PLL system has started and some promising results have been obtained.

APPENDIX A

The Mathematical Model of the Induction Machine

A.1 Induction Machines Space Vector Representation

To find the mathematical model of Induction machines, it is useful to represent the machine electrical quantities as space vectors. In the following equations the space vector representation of the stator currents, voltages and fluxes are defined in the stationary reference frame:

$$\bar{i}_s = \frac{2}{3}(i_{sA} + ai_{sB} + a^2 i_{sC}) \tag{A.1}$$

$$\bar{i}_s = i_{sD} + j \cdot i_{sQ}$$

$$\bar{v}_s = \frac{2}{3}(v_{sA} + a \cdot v_{sB} + a^2 \cdot v_{sC}) \tag{A.2}$$

$$\bar{v}_s = v_{sD} + j \cdot v_{sQ}$$

$$\bar{\psi}_s = \frac{2}{3}(\psi_{sA} + a \cdot \psi_{sB} + \psi^2 \cdot i_{sC}) \tag{A.3}$$

$$\bar{\psi}_s = \psi_{sD} + j \cdot \psi_{sQ}$$

Where

$$a = e^{j\frac{2\pi}{3}} = -\frac{1}{2} + j\frac{\sqrt{3}}{2} \tag{A.4}$$

$$a^2 = e^{j\frac{4\pi}{3}} = -\frac{1}{2} - j\frac{\sqrt{3}}{2} \tag{A.5}$$

The voltage equations of three phase machines are written as:

$$v_{sA} = R_s i_{sA} + \frac{d\psi_{sA}}{dt} \tag{A.6}$$

$$v_{sB} = R_s i_{sB} + \frac{d\psi_{sB}}{dt} \tag{A.7}$$

$$v_{sC} = R_s i_{sC} + \frac{d\psi_{sC}}{dt} \tag{A.8}$$

Substituting these equations in the voltage space vector equation (A.2) gives:

$$\bar{v}_s = \frac{2}{3}\left(\left(R_s i_{sA} + \frac{d\psi_{sA}}{dt}\right) + a\left(R_s i_{sB} + \frac{d\psi_{sB}}{dt}\right) + a^2\left(R_s i_{sC} + \frac{d\psi_{sC}}{dt}\right)\right) \tag{A.9}$$

$$\bar{v}_s = \frac{2}{3}\left(R_s\left(i_{sA} + a i_{sB} + a^2 i_{sC}\right) + \left(\frac{d\psi_{sA}}{dt} + a\frac{d\psi_{sB}}{dt} + a^2\frac{d\psi_{sC}}{dt}\right)\right) \tag{A.10}$$

Substituting (A.1) and (A.3) in (A.10) yields the stator voltage equation in the space vector representation:

$$\bar{v}_s^s = \frac{2}{3}\left(R_s \cdot \bar{i}_s^s + \frac{d\bar{\psi}_s^s}{dt}\right) \tag{A.11}$$

In (A.11), the subscript s and the superscript s denote the stator and the stator reference frame respectively.

The electrical quantities in the rotor can be represented as space vectors in the same way as the stator quantities in a rotating reference frame fixed to the rotor as the following:

$$\bar{i}_r = \frac{2}{3}(i_{rA} + a \cdot i_{rB} + a^2 \cdot i_{rC}) \tag{A.12}$$

$$\bar{v}_r = \frac{2}{3}(v_{rA} + a \cdot v_{rB} + a^2 \cdot v_{rC}) \tag{A.13}$$

$$\overline{\psi}_r = \frac{2}{3}(\psi_{rA} + a \cdot \psi_{rB} + \psi^2 \cdot i_{rC}) \tag{A.14}$$

In a three phase machine, the rotor voltage equations can be written as:

$$v_{rA} = R_r i_{rA} + \frac{d\psi_{rA}}{dt} \tag{A.15}$$

$$v_{rB} = R_r i_{rB} + \frac{d\psi_{rB}}{dt} \tag{A.16}$$

$$v_{rC} = R_r i_{rC} + \frac{d\psi_{rC}}{dt} \tag{A.17}$$

Substituting (A.15), (A.16) and (A.17) into (A.13) yields:

$$\bar{v}_r = \frac{2}{3}\left(\left(R_r i_{rA} + \frac{d\psi_{rA}}{dt}\right) + a\left(R_r i_{rB} + \frac{d\psi_{rB}}{dt}\right) + a^2\left(R_r i_{rC} + \frac{d\psi_{rC}}{dt}\right)\right) \tag{A.18}$$

$$\bar{v}_r = \frac{2}{3}\left(R_r\left(i_{rA} + a i_{rB} + a^2 i_{rC}\right) + \left(\frac{d\psi_{rA}}{dt} + a\frac{d\psi_{rB}}{dt} + a^2\frac{d\psi_{rC}}{dt}\right)\right) \tag{A.19}$$

From (A.12) and (A.14) the rotor voltage equation in the space vector form can be written as:

$$\bar{v}_r^r = \frac{2}{3}\left(R_r \cdot \bar{i}_r^r + \frac{d\overline{\psi}_r^r}{dt}\right) \tag{A.20}$$

In (A.20), the subscript r and the superscript r denote the rotor quantities and the rotor reference frame respectively.

In order to find the mathematical model of the induction machines, all the machine equations should be written in the same reference frame. Therefore in the next section, the transformation between the different reference frames is discussed.

A.2 Reference Frame Transformation

The first transformation considered is between the stator reference (*D-Q*) frame and a rotating reference frame (*d-q*) Fig. A.1.

From the figure the following equations can be found:

$$i_{sD} = i_{sd}\cos\theta_e - i_{sq}\sin\theta_e \tag{A.21}$$

$$i_{sQ} = i_{sd}\sin\theta_e + i_{sq}\cos\theta_e \tag{A.22}$$

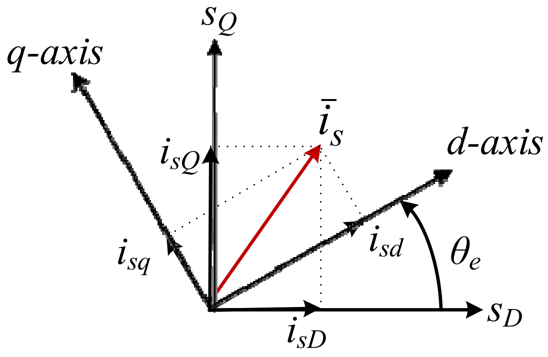


Fig. 0.1 Transformation from the stationary to the synchronous frames

In the space vector representation, the transformation can be written as:

$$\bar{i}_s^s = i_{sD} + ji_{sQ} = \bar{i}_s^e e^{j\theta_e} = (i_{sd} + ji_{sq})e^{j\theta_e} \tag{A.23}$$

The superscript *e* denotes the (*d-q*) reference frame.

The transformation can be written also in matrix form as:

$$\begin{bmatrix} i_{sD} \\ i_{sQ} \end{bmatrix} = \begin{bmatrix} \cos\theta_e & -\sin\theta_e \\ \sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} i_{sd} \\ i_{sq} \end{bmatrix} \tag{A.24}$$

Similarly, the inverse transformation can be written as:

$$\bar{i}_s^e = i_{sd} + ji_{sq} = \bar{i}_s^s e^{-j\theta_e} = (i_{sD} + ji_{sQ})e^{-j\theta_e} \tag{A.25}$$

And in matrix for:

$$\begin{bmatrix} i_{sd} \\ i_{sq} \end{bmatrix} = \begin{bmatrix} \cos\theta_e & \sin\theta_e \\ -\sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} i_{sD} \\ i_{sQ} \end{bmatrix} \tag{A.26}$$

In a similar way, the stator electrical quantities can be written in a reference frame fixed to the rotor as:

$$\bar{i}_s^r = \bar{i}_s^s e^{-j\theta_r} \tag{A.27}$$

$$\bar{v}_s^r = \bar{v}_s^s e^{-j\theta_r} \tag{A.28}$$

$$\bar{\psi}_s^r = \bar{\psi}_s^s e^{-j\theta_r} \tag{A.29}$$

and the rotor quantities can be written in a stationary reference frame as:

$$\bar{i}_r^s = \bar{i}_r^r e^{j\theta_r} \tag{A.30}$$

$$\bar{v}_r^s = \bar{v}_r^r e^{j\theta_r} \tag{A.31}$$

$$\bar{\psi}_r^s = \bar{\psi}_r^r e^{j\theta_r} \tag{A.32}$$

A.3 The Induction Machine Mathematical Model

To find the mathematical model of the induction machine, the rotor voltage equation in the stationary reference frame is found by substituting (A.30), (A.31) and (A.32) into (A.20) this yields:

$$\bar{v}_r^s e^{-j\theta_r} = R_r \bar{i}_r^s e^{-j\theta_r} + \frac{d}{dt}\left(\bar{\psi}_r^s e^{-j\theta_r}\right) \tag{A.33}$$

By defining $\omega_r = \frac{d\theta_r}{dt}$ which is the rotor angular speed, (A.33) becomes:

$$\bar{v}_r^s e^{-j\theta_r} = R_r \bar{i}_r^s e^{-j\theta_r} + e^{-j\theta_r}\frac{d\bar{\psi}_r^s}{dt} - j\omega_r \bar{\psi}_r^s e^{-j\theta_r} \tag{A.34}$$

By eliminating $e^{-j\theta_r}$ from both the equation sides, the rotor voltage equation in a stationary reference frame can be expressed as:

$$\bar{v}_r^s = R_r \bar{i}_r^s + \frac{d\bar{\psi}_r^s}{dt} - j\omega_r \bar{\psi}_r^s \tag{A.34}$$

(A.10) and (A.34) represent the induction machine mathematical model represented in a stationary reference frame and which can also be written in term of its real and imaginary parts as:

$$v_{sD} = R_s i_{sD} + \frac{d\psi_{sD}}{dt} \tag{A.35}$$

$$v_{sQ} = R_s i_{sQ} + \frac{d\psi_{sQ}}{dt} \tag{A.36}$$

$$v_{rd} = R_r i_{rd} + \frac{d\psi_{rd}}{dt} + \omega_r \psi_{rq} \tag{A.37}$$

$$v_{rq} = R_r i_{rq} + \frac{d\psi_{rq}}{dt} + \omega_r \psi_{rd} \tag{A.38}$$

The linkage fluxes in the stator and rotor can be expressed in terms of stator and rotor currents as:

$$\bar{\psi}_s = L_s \bar{i}_s + L_m \bar{i}_r \tag{A.39}$$

$$\bar{\psi}_r = L_m \bar{i}_s + L_r \bar{i}_r \tag{A.40}$$

where L_m is the mutual inductance and L_s, L_r are the stator and rotor self-inductances respectively and they are equal:

$$L_s = L_{ls} + L_m \tag{A.41}$$

$$L_r = L_{lr} + L_m \tag{A.42}$$

where L_{ls} and L_{lr} are the stator and rotor leakage inductances respectively.

(A.39) and (A.40) can be expressed also as:

$$\psi_{sD} = L_s i_{sD} + L_m i_{rd} \tag{A.43}$$

$$\psi_{sQ} = L_s i_{sQ} + L_m i_{rq} \tag{A.44}$$

$$\psi_{rd} = L_m i_{sD} + L_r i_{rd} \tag{A.45}$$

$$\psi_{rq} = L_m i_{sQ} + L_r i_{rq} \tag{A.46}$$

Substituting (A.39) into (A.10) gives:

$$\bar{v}_s^s = \left(R_s + L_s p\right)\bar{i}_s^s + L_m p \bar{i}_r^s \tag{A.47}$$

and substituting (A.40) into (A.34) gives:

$$\bar{v}_r^s = \left(R_r + L_r p\right)\bar{i}_r^s + L_m p \bar{i}_s^s - j\omega_r\left(L_m \bar{i}_s^s + L_r \bar{i}_r^s\right) \tag{A.48}$$

where p denotes $\dfrac{d}{dt}$ the differential operator.

The induction machine voltage equations can be written in matrix fom as:

$$\begin{bmatrix} \bar{v}_s^s \\ \bar{v}_r^s \end{bmatrix} = \begin{bmatrix} R_s + L_s p & L_m p \\ (p - j\omega_r)L_m & R_r + (p - j\omega_r)L_r \end{bmatrix} \begin{bmatrix} \bar{i}_s^s \\ \bar{i}_r^s \end{bmatrix} \tag{A.49}$$

Equations (A.47) and (A.48) can be also written as:

$$v_{sD} = \left(R_s + L_s p\right) i_{sD} + L_m p i_{rd} \tag{A.50}$$

$$v_{sQ} = \left(R_s + L_s p\right) i_{sQ} + L_m p i_{rq} \tag{A.51}$$

$$v_{rd} = L_m p i_{sD} + \omega_r L_m i_{sQ} + \left(R_r + L_r p\right) i_{rd} + \omega_r L_r i_{rq} \tag{A.52}$$

$$v_{rq} = -\omega_r L_m i_{sD} + L_m p i_{sQ} - \omega_r L_r i_{rd} + \left(R_r + L_r p\right) i_{rq} \tag{A.53}$$

A.4 The Induction Machine State Space Model

The state space of the induction machine is developed in this section, where the states are the stator currents and the rotor fluxes in a stationary reference frame. The rotor current space vector can be written in terms of the stator current and rotor flux vectors as:

$$\bar{i}_r = \frac{1}{L_r}(\bar{\psi}_r - L_m \bar{i}_s) \tag{A.54}$$

Substituting (A.54) in (A.47) gives:

$$\bar{v}_s = \left(R_s + L_s p\right)\bar{i}_s + \frac{L_m}{L_r} p(\bar{\psi}_r - L_m \bar{i}_s) \tag{A.55}$$

$$\bar{v}_s = R_s \bar{i}_s + \left(L_s - \frac{L_m^2}{L_r}\right) p \bar{i}_s + \frac{L_m}{L_r} p \bar{\psi}_r \tag{A.56}$$

$$\bar{v}_s = R_s \bar{i}_s + L_s \left(1 - \frac{L_m^2}{L_s L_r}\right) p \bar{i}_s + \frac{L_m}{L_r} p \bar{\psi}_r \tag{A.57}$$

By defining the leakage coefficient σ as:

$$\sigma = 1 - \frac{L_m^2}{L_s L_r} \tag{A.58}$$

The voltage equation becomes:

$$\bar{v}_s = R_s \bar{i}_s + \sigma L_s p \bar{i}_s + \frac{L_m}{L_r} p \bar{\psi}_r \tag{A.59}$$

(A.59) can be written in terms of (*D*-*Q*) as:

$$v_{sD} = R_s i_{sD} + \sigma L_s p i_{sD} + \frac{L_m}{L_r} p \psi_{rd} \tag{A.60}$$

$$v_{sQ} = R_s i_{sQ} + \sigma L_s p i_{sQ} + \frac{L_m}{L_r} p \psi_{rq} \tag{A.61}$$

Also substituting (A.54) in (A.48) gives:

$$0 = \frac{1}{T_r}\left(\overline{\psi}_r - L_m \bar{i}_s\right) + \frac{d\overline{\psi}_r}{dt} - j\omega_r \overline{\psi}_r \tag{A.62}$$

where T_r is the rotor time constant and is defined as:

$$T_r = \frac{R_r}{L_r} \tag{A.63}$$

Rearranging (A.59) gives:

$$p\overline{\psi}_r = \frac{L_m}{T_r}\bar{i}_s - \frac{1}{T_r}\overline{\psi}_r + j\omega_r \overline{\psi}_r \tag{A.64}$$

Or in terms of the (*D*-*Q*) coordinates:

$$p\psi_{rd} = \frac{L_m}{T_r} i_{sD} - \frac{1}{T_r}\psi_{rd} - \omega_r \psi_{rq} \tag{A.65}$$

$$p\psi_{rq} = \frac{L_m}{T_r} i_{sQ} - \frac{1}{T_r}\psi_{rq} - \omega_r \psi_{rd} \tag{A.66}$$

Substituting (A.65) and (A.66) into (A.60) and (A.61) yields:

$$v_{sD} = R_s i_{sD} + \sigma L_s p i_{sD} + \frac{L_m}{L_r} p\left(\frac{L_m}{T_r} i_{sD} - \frac{1}{T_r}\psi_{rd} - \omega_r \psi_{rq}\right) \tag{A.67}$$

$$v_{sQ} = R_s i_{sQ} + \sigma L_s p i_{sQ} + \frac{L_m}{L_r} p\left(\frac{L_m}{T_r} i_{sD} - \frac{1}{T_r}\psi_{rd} - \omega_r \psi_{rq}\right) \tag{A.68}$$

Separating the derivative terms in (A.67) and (A.68) gives:

$$pi_{sD} = \frac{v_{sD}}{\sigma L_s} - \left(\frac{R_s}{\sigma L} + \frac{1-\sigma}{\sigma T_r} \right) i_{sD} + \frac{L_m}{\sigma L_s L_r T_r} \psi_{rd} + \frac{L_m}{\sigma L_s L_r} \omega_r \psi_{rq} \tag{A.69}$$

$$pi_{sQ} = \frac{v_{sQ}}{\sigma L_s} - \left(\frac{R_s}{\sigma L} + \frac{1-\sigma}{\sigma T_r} \right) i_{sQ} + \frac{L_m}{\sigma L_s L_r T_r} \psi_{rq} - \frac{L_m}{\sigma L_s L_r} \omega_r \psi_{rd} \tag{A.70}$$

(A.65), (A.66), (A.69) and (A.70) can be used to find the state space model representing the induction machine in a stationary reference frame as:

$$\begin{bmatrix} pi_{sD} \\ pi_{sQ} \\ p\psi_{rd} \\ p\psi_{rq} \end{bmatrix} = \begin{bmatrix} -a_1 & 0 & a_2 & a_3\omega_r \\ 0 & -a_1 & -a_3\omega_r & a_2 \\ \frac{L_m}{T_r} & 0 & -\frac{1}{T_r} & -\omega_r \\ 0 & \frac{L_m}{T_r} & \omega_r & -\frac{1}{T_r} \end{bmatrix} \begin{bmatrix} i_{sD} \\ i_{sQ} \\ \psi_{rd} \\ \psi_{rq} \end{bmatrix} + \begin{bmatrix} \frac{1}{\sigma L_s} & 0 \\ 0 & \frac{1}{\sigma L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_{sD} \\ v_{sQ} \end{bmatrix} \tag{A.71}$$

where:

$$a_1 = \frac{R_s}{\sigma L} + \frac{1-\sigma}{\sigma T_r} \tag{A.72}$$

$$a_2 = \frac{L_m}{\sigma L_s L_r T_r} \tag{A.73}$$

$$a_3 = \frac{L_m}{\sigma L_s L_r} \tag{A.74}$$

A.5 Power and Torque Equations

The input power of the induction machines can be written in terms of the three phase voltages and currents as:

$$P_i = v_{sA} i_{sA} + v_{sB} i_{sB} + v_{sC} i_{sC} \tag{A.75}$$

By substituting the three phase quantities by their equivalent two phase quantities, (A.75) becomes:

$$P_i = v_{sD} i_{sD} + \left(-\frac{1}{2} v_{sD} + \frac{\sqrt{3}}{2} v_{sQ} \right)\left(-\frac{1}{2} i_{sD} + \frac{\sqrt{3}}{2} i_{sQ} \right) + \\ + \left(-\frac{1}{2} v_{sD} - \frac{\sqrt{3}}{2} v_{sQ} \right)\left(-\frac{1}{2} i_{sD} - \frac{\sqrt{3}}{2} i_{sQ`} \right) \tag{A.76}$$

$$P_i = v_{sD} i_{sD} + \frac{1}{4} v_{sD} i_{sD} - \frac{\sqrt{3}}{4} v_{sD} i_{sQ} - \frac{\sqrt{3}}{4} v_{sQ} i_{sD} + \frac{3}{4} v_{sQ} i_{sQ} + \\ + \frac{1}{4} v_{sD} i_{sD} + \frac{\sqrt{3}}{4} v_{sD} i_{sQ} + \frac{\sqrt{3}}{4} v_{sQ} i_{sD} + \frac{3}{4} v_{sQ} i_{sQ} \tag{A.77}$$

$$P_i = \frac{3}{2}\left(v_{sD} i_{sD} + v_{sQ} i_{sQ} \right) \tag{A.78}$$

The mechanical power at the machine shaft can be calculated from:

$$P_{mech} = \frac{3}{2}\left(e_{rd} i_{rd} + e_{rq} i_{rq} \right) \tag{A.79}$$

where $\overline{e}_r$ is the rotor EMF space vector which can be written as:

$$\overline{e}_r = e_{rd} + j e_{rq} \tag{A.80}$$

$$\overline{e}_r = -j\omega_r \overline{\psi}_r = -j\omega_r \left(\psi_{rd} + j\psi_{rq} \right) \tag{A.81}$$

$$\overline{e}_r = \omega_r \psi_{rq} - j\omega_r \psi_{rd} \tag{A.82}$$

Substituting (A.82) into (A.79) yields:

$$P_{mech} = \frac{3}{2}\omega_r \left(\psi_{rq} i_{rd} - \psi_{rd} i_{rq} \right) \tag{A.83}$$

The electromechanical torque can be calculated from the mechanical power equation by applying:

$$T_e = \frac{P_{mech}}{\omega_{rm}} = \frac{3}{2} P\left(\psi_{rq} i_{rd} - \psi_{rd} i_{rq} \right) = -\frac{3}{2} P\left(\overline{\psi}_r \times \overline{i}_r \right) \tag{A.84}$$

where P is the number of the pole pairs and ω_{rm} is the mechanical angular rotor speed which can be found from the electrical rotor angular speed by applying:

$$\omega_{rm} = \frac{\omega_r}{P} \tag{A.85}$$

By applying the same way, the torque equation in terms of other machine variables can be found as:

$$T_e = -\frac{3}{2}P\left(\bar{\psi}_r \times \bar{i}_r\right) = \frac{3}{2}P\left(\psi_{rq} i_{rd} - \psi_{rd} i_{rq}\right) \tag{A.86}$$

$$T_e = -\frac{3}{2}PL_m\left(\bar{i}_s \times \bar{i}_r\right) = \frac{3}{2}PL_m\left(i_{sQ} i_{rd} - i_{sD} i_{rq}\right) \tag{A.87}$$

$$T_e = -\frac{3}{2}P\frac{L_m}{L_r}\left(\bar{i}_s \times \bar{\psi}_r\right) = \frac{3}{2}P\frac{L_m}{L_r}\left(\psi_{rd} i_{sQ} - \psi_{rq} i_{sD}\right) \tag{A.88}$$

APPENDIX B

The Algorithm C-code

```
//################################################################################
//
// FILE:    F28335_drive_main.c
// Author:  Yaman Zbede
// Date: 07-05-14
//
// TITLE:  Model Predictive MRAS Estimator
//
//    Features:
//    Configured for "boot to SARAM" operation.
//             10kHz PWM on 6 channels
//    Primary control code runs in epwm1_isr
//    Encoder interfaced via Eqep1
//    User interface using Labview GUI (RS232 via SCIA)
//    4-channel DAC (SPIA)
//    Eqep1 interrupt for encoder watchdog and direction reversal detection
//
//    Changed LSPCLK to 75MHz to increase SPI-A clk to 18.75MBaud to effect faster transfers to DAC
//    This was done in DSP2833x_SysCtrl.c
//    SCIA (RS232) baud set to 585937.

//
#define CURRENT_LOOP_INT_CLAMP       3750      // Integrator clamp for antiwnd-up
#define SPEED_LOOP_INT_CLAMP         8
#define iqe_demand_CLAMP          8
#define POS_LOOP_INT_CLAMP           1000
#define PI 3.141592653589
#define TABLEN  720                                         // Set length of sine/cosine look-up tables
#define   DATA_STORE_LEN             0x800          // Set length of data store records to 2k
#define Rs 2.35
#define Rr 1.05
#define Ls 0.344209
#define Lr 0.348197
#define Lm 0.33209
#define sigma 1-(Lm*Lm/Ls/Lr)
#define Ts 0.0001
#define aa Lr/Lm*0.57735/3750
#define bb Lr/Lm*Rs
#define cc Lr/Lm*sigma*Ls/Ts
#define Tr Lr/Rr
#define dd Ts*Lm/Tr
#define ee 1-Ts/Tr




// Sensor calibration settings
// The gain from CAS15-NP primary current to ADC reading was measured at 100 per ampere
// 1/100 = 0.01
#define SENSOR_GAIN0  0.01                  // Sensor gain (CAS15-NP) measured in ampere
#define SENSOR_GAIN1  0.01                  // Sensor gain (CAS15-NP) measured in ampere
#define SENSOR_GAIN2  0.01                  // Sensor gain (CAS15-NP) measured in ampere
```

```
//#define SENSOR_GAIN0 10                          // Sensor gain (CAS15-NP) mA
//#define SENSOR_GAIN1 10                          // Sensor gain (CAS15-NP) mA
//#define SENSOR_GAIN2 10                          // Sensor gain (CAS15-NP) Ma
#define Speed_Rep       200
#define speed_div       4.175138628/2   //500 sample
//#define speed_div       0.4158271737     //500 sample
//#define speed_div       0.4158271737*0.5    //1000 sample

#define SENSOR_GAIN3  0.0                                  // Sensor gain
#define SENSOR_GAIN4  0.0                                  // Sensor gain
#define SENSOR_GAIN5  0.0                                  // Sensor gain
#define SENSOR_OFFSET0         2039                        // Sensor offset (CAS15-NP)
#define SENSOR_OFFSET1         2042                        // Sensor offset (CAS15-NP)
#define SENSOR_OFFSET2         2026                        // Sensor offset (CAS15-NP)
#define SENSOR_OFFSET3         0                                   // Sensor offset
#define SENSOR_OFFSET4         0                                   // Sensor offset
#define SENSOR_OFFSET5         0                                   // Sensor offset
// The following #defines are related to the way the Eqep calculates encoder velocity
//#define VELOCITY_SCALE                2197256// Scale for CCPS = 4, UPPS = 5
//#define VELOCITY_SCALE                1098628// Scale for CCPS = 5, UPPS = 5
//#define VELOCITY_SCALE                274657  // Scale for CCPS = 7, UPPS = 5
#define VELOCITY_SCALE                  137328  // Scale for CCPS = 7, UPPS = 4
#include "DSP2833x_Device.h"                    // DSP2833x Headerfile Include File
#include "DSP2833x_Examples.h"                  // DSP2833x Examples Include File
#include "F28335_drive_function_prototypes.h"   // F28335 drive header file (DJA)
#include <stdlib.h>                                        // needed for calloc()
#include <math.h>
#include <string.h>
#include <stdio.h>

// Function prototypes for functions defined in this file
// Code located in external RAM
void set_up_data_stores(void);
void set_up_look_up_tables(void);
void panel_controls(void);
// Code located in internal RAM
void update_panel(void);
void transfer_store_data_to_RS232(int frame_count);

// Interrupt service routines defined in this file
interrupt void epwm1_isr(void);        // This ISR contains application control code
interrupt void eqep1_isr(void);                  // This ISR is used for diagnostic only

// Global variables
float kp_alpha=0,ki_alpha=0,valpha=0,vbeta=0,zd_alpha=0,zd_beta=0,error_alpha=0,error_beta=0;
long start_time=0;
int theta_r_1=0;
float Fai_rr_d_1 = 0, Fai_rr_q_1 = 1;   //Rotor Flux in Reference Model
float speed_new=0,speed_new_1=0;        //speed with low pass filter
float error_MRAS0=0,error_MRAS1=0,error_MRAS2=0,error_MRAS3=0;
int *s1, *s2, *s3, *s4, *s5, *s6, *s7, *s8;          // pointers for external RAM data store
float sin_t[TABLEN], cos_t[TABLEN];              // look-up tables (code quicker when declared in INT RAM)
int imod = 0, vmod = 0;                                                          // modulus of current
vector
int reverse = -1;            // when -1 the motor spin in the opposite direction
Uint16 flag1 = 0, flag2 = 0, flag3 = 0, flag4 = 0, flag5 = 0;
Uint16 pb1, pb2, pb3, pb4, pb5, pb6, pb7;
int B0 = 0, B1 = 0, B2 = 0, B3 = 0, B4 = 0, B5 = 0, B6 = 0, B7 = 0;
int B8 = 0, B9 = 0, B10 = 0, B11 = 0, B12 = 0, B13 = 0, B14 = 0, B15 = 0;
int par1 = 0, par2 = 0, par3 = 0, par5 = 0, par6 = 0;// Parameters passed in from control panel
int par1b = 0;                                             // used in code for DESAT single pulse test
Uint16 par4 = 0;                                 // use unsigned as it houses individual bits
Uint16 xcount = 0;                                                               // cycle counter
Uint16 theta_e = 0;                                                              // voltage vector angle
Uint16 theta_r = 0;                                                              // voltage vector angle
int i=0;
float sum=0;
float theta_e2 = 0;                                                     // spare angle for testing
purposes
float theta_e3 = 4000;
Uint16 theta_e4 = 0;
Uint32 Speed_count;
float chopper = 0;
float change = 0;
Uint16 store_enable = 0;                                                // data store enable flag
Uint16 store_counter = 0;                                         // data store index
unsigned long ISR_count1 = 0;       // ISR counter (increments every ISR execution)
unsigned long LoopCount = 0;                   // LabVIEW (RS232) data transfer counter
long int temp_count = 0;                                                // event delay counter
Uint16 ErrorCount;                                                            // used by RS232 code
Uint16 pos_count = 0;                                                         // Encoder position
count
Uint16 QCTMR_count = 0;                                                 // Encoder velocity timer count
int N_RPM = 0;
int N_RPM_E = 0;
int N_RPM1 = 0;                                                   // Shaft velocity in RPM
float Pos_Yam = 0;
float Position = 0;
static float Pos_Yam_1 = 0;
Uint16 reversal_count = 0;                                        // Speed reversal counter
Uint16 release = 0;                                               // Integrator release (0=hold, 1=run)
Uint16 enc_wdog = 0;                                              // EQep1 watchdog timeout counter
Uint16 QCTMR_overflow = 0;
float acc1 = 0, acc2 = 0, acc3 = 0;   // These variables are used for calculating averages
int ave1 = 0, ave2 = 0, ave3 = 0;
Uint16 T0_count = 0;                // Initial Timer0 (used for timing code in us)
Uint16 TZflag = 0;                                     // Trip Zone status (0=clear, 1=trip)
Uint16 trip_count = 0;                                                  // CBC trip counter
Uint16 countA = 0;                                                            // isr counter
Uint16 flag = 0;                                                        // pulse on flag
Uint16 comp2 = 0;                                                                   // comp2
button status
Uint16 comp2p = 0;                                                            // previous comp2
button status
Uint16 maxcount = 0;                                   // number of PWM cycles in test pulse
int disp1, disp2;

// The following flags maintain the status of the front panel buttons
Uint16 Clear_PWM_trip_button_status = 0;
Uint16 Relay1_button_status = 0;
Uint16 Relay2_button_status = 0;
Uint16 Relay3_button_status = 0;
Uint16 Relay4_button_status = 0;
Uint16 Force_PWM_trip_button_status = 0;
Uint16 Data_Store_button_status = 0;
Uint16 Comp_2_button_status = 0;

// The following flags maintain the status of the relays
Uint16 Relay1_status = 0;
Uint16 Relay2_status = 0;
Uint16 Relay3_status = 0;
Uint16 Relay4_status = 0;

// These following global variables are used to transfer data stored in ISR local variables to other functions
int vde_m = 0, vqe_m = 0, ide_m = 0, iqe_m = 0;

int main(void) {
        Uint16 ReceivedChar = 0;
        char sbuf[100];                 // character buffer for LabVIEW data link (RS232)
        char letter[100], temp = 0;
        Uint16 i = 0, nc = 0;

        set_up_data_stores();           // set up and initialise data stores in ext RAM
        set_up_look_up_tables();                // set up sine and cosine look-up tables
```

```
// Clear RS232 input string buffer
        for (i = 0; i < 100; ++i)
                letter[i] = 0;

// Initialise System Control:
// PLL, WatchDog, enable Peripheral Clocks
// This example function is found in the DSP2833x_SysCtrl.c file.
        InitSysCtrl();
// Initialise the pins for the SCI-A port (RS232).
        InitSciaGpio();

// Initialise the pins for the SPI-A port (DAC).
        InitSpiaGpio();

// Initialise GPIO:
// This example function is found in the DSP2833x_Gpio.c file and
// illustrates how to set the GPIO to it's default state.
// InitGpio();  // Skipped for this application

// Initialise GPIO pins for ePWM1, ePWM2, ePWM3, ePWM4, ePWM5, ePWM6
// These functions are in the DSP2833x_EPwm.c file
        InitEPwm1Gpio();
        InitEPwm2Gpio();
        InitEPwm3Gpio();
        InitEPwm4Gpio();
        InitEPwm5Gpio();
        InitEPwm6Gpio();

// Initialise GPIO pins or EQEP (code added on 29-03-09)
// This function is in DSP2833x_EQep.c
// This function configures both EQEP1 and EQEP2
        InitEQep1Gpio();

// Initialise GPIO (gate drive reset, relays, DAC and test points)
        gpio_init();

// Clear all interrupts and initialise PIE vector table:
// Disable CPU interrupts
        DINT;

// Initialise the PIE control registers to their default state.
// The default state is all PIE interrupts disabled and flags
// are cleared.
// This function is found in the DSP2833x_PieCtrl.c file.
        InitPieCtrl();

// Disable CPU interrupts and clear all CPU interrupt flags:
        IER = 0x0000;
        IFR = 0x0000;

// Initialise the PIE vector table with pointers to the shell Interrupt
// Service Routines (ISR).
// This will populate the entire table, even if the interrupt
// is not used in this example.  This is useful for debug purposes.
// The shell ISR routines are found in DSP2833x_DefaultIsr.c.
// This function is found in DSP2833x_PieVect.c.
        InitPieVectTable();

// Interrupts that are used in this example are re-mapped to
// ISR functions found within this file.

        EALLOW;
        // This is needed to write to EALLOW protected registers
        PieVectTable.ADCINT = &epwm1_isr;          // ADC triggered ISR
        PieVectTable.EQEP1_INT = &eqep1_isr;       // EQEP1 triggered ISR
        EDIS;
        // This is needed to disable write to EALLOW protected registers

// Set up ADC (function below is located in DSP2833x_Adc.c)
// The following function sets ADCCLK to 37.5MHz if CPS = 1 (this is too high as ADCCLK <= 25MHz)
// Need to adjust ADCTRL3 to set to 25MHz
        InitAdc();

// Initialise ADC sequencer
        adc_seq_init();

        EALLOW;
        SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 0;          // ePWM TBCLK stopped
        EDIS;

// Initialise PWM modules
        InitEPwmMods();

        EALLOW;
        SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC = 1;// All enabled ePWMs synchronised with rising edge of
TBCLK
        EDIS;

// Set up Timer0 for code execution time measurement within epwm1_isr
        Timer0_init();

// Set up SPI for DAC
        spi_fifo_init();    // Initialise the SPI FIFO
        spi_init();                 // initialise SPI

// Set up Eqep1 for encoder position and speed measurement
        eqep1_init();

// Enable CPU interrupts
        IER |= M_INT1;// Enable CPU INT1 which is connected to PIE group 1 (main control ISR)
        IER |= M_INT5;// Enable CPU INT5 which is connected to PIE group 5 (shaft direction change ISR)

        PieCtrlRegs.PIEIER1.bit.INTx6 = 1;// Enable ADCINT in the PIE: Group 1 interrupt 6 (i.e. INT1.6)
        PieCtrlRegs.PIEIER5.bit.INTx1 = 1;// Enable EQEP1_INT in the PIE: Group 5 interrupt 1 (i.e. INT5.1)

// Enable global Interrupts and higher priority real-time debug events:
        EINT;
        // Enable Global interrupt INTM
        ERTM;
        // Enable Global real-time interrupt DBGM

        scia_fifo_init();     // Initialise the SCI FIFO
        scia_echoback_init();  // Initialise SCI for echoback

        LoopCount = 0;                  // Set LabVIEW data transfer counter to zero
        ErrorCount = 0;
        i = 0;                                  // Set RS232 character counter to zero

// IDLE loop. Just sit and loop forever:
// This loop is concerned only with communications with LabVIEW
        for (;;) {
                i = 0;      //reset input character counter
                do {
                        // Wait for incomin g character from RS232 port
                        while (SciaRegs.SCIFFRX.bit.RXFFST != 1) {
                        }           // wait for XRDY =1 for empty state
                        ReceivedChar = SciaRegs.SCIRXBUF.all;           // Get character
                        temp = ReceivedChar & 0xFF;                             // strip off the error bits
                        letter[i] = temp;
                        i++;
                } while (temp != '\n');

                // Get incoming parameters from LabVIEW GUI
                nc = sscanf(letter, "%d %d %d %d %d %d", &par1, &par2, &par3, &par4,
                                &par5, &par6);

                // Test for sensor out-of-range trip (send status to LabVIEW)
                TZflag = EPwm1Regs.TZFLG.bit.OST;

                // Inspect par1 to determine LabVIEW interface mode
                // par1=200 means that LabVIEW is in User Control Panel (UCP) mode
                // par1=100 means that LabVIEW in in Data Transfer Interface (DTI) mode

                // Send data to LabVIEW (UCP mode)
                if (par1 == 200)
                        update_panel();
                // Send data to LabVIEW (DTI mode)
                if (par1 == 100)
                        transfer_store_data_to_RS232(par2);

                get_control_panel_button_status();// Update LabVIEW panel push button status flags

                panel_controls();  // Implement the pushbutton commands

                LoopCount++;                    // Count LabVIEW data transfer cycles
        }
        return 0;
} // end main()

//----------------------------------------------------------------------------------------------------
// This primary ISR is used for the converter control and is synchronised to EPWM1 (1st PWM channel)
// The PWM triggers the ADC sequencer. When the sequencer is finished it triggers this ISR
// The ADC SEQ1 is triggered on the positive apex of the carrier
// The CPU INT1.6 is triggered on end of sequence (EOS)

interrupt void epwm1_isr(void) {
        float vds, vqs, vde = 0, vqe = 0;// stator voltages, 2-axis, stator and e-frame
        float va, vb, vc;                                       // stator phase voltage demands
        float ia = 0, ib = 0, ic = 0;                   // measured stator phase currents
        float i4 = 0, i5 = 0, i6 = 0;                           // other currents
        float ids, iqs;                                 // measured stator currents (stator ref frame)
        static float ids_1 = 0, iqs_1 = 0;// measured stator currents (stator ref frame)
        float Sai_r_d, Sai_r_q;             //Rotor Flux in Reference Model
        float Fai_r_d, Fai_r_q;             //Rotor Flux in adaptive Model
        float Fai_rr_d, Fai_rr_q;             //Rotor Flux in adaptive Model (Rotor Reference Frame)
        static float Sai_r_d_1 = 0, Sai_r_q_1 = 0;   //Rotor Flux in Reference Model
        static float Sai_r_d_2 = 0, Sai_r_q_2 = 0;   //Rotor Flux in Reference Model
        static float Sai_r_d_3 = 0, Sai_r_q_3 = 0;   //Rotor Flux in Reference Model
        static float Sai_r_d_4 = 0, Sai_r_q_4 = 0;   //Rotor Flux in Reference Model
        static float Sai_r_d_5 = 0, Sai_r_q_5 = 0;   //Rotor Flux in Reference Model
        float error_MRAS = 0, error_MRAS_1 = 0;  //Rotor Flux MRAS errror
        float ide, iqe;                                         // measured stator currents (e-frame)
        float idr, iqr;                                         // measured stator currents (r-frame)
Yaman Zbede
        float ide_demand, iqe_demand;                   // d and q axis current demands
        float id_error, iq_error;                       // d and q axis current errors
        float kp, ki;                                           // current loop controller gains
        float kps, kis;                                         // speed loop controller gains
        float kpp, kip;                                         // position loop controller gains
        static float zd = 0, zq = 0;            // current loop integrator accumulators
        static float zs = 0;                                    // speed loop integrator accumulator
        static float zp = 1;                            // position loop integrator accumulator
        float vide_demand, viqe_demand;                 // voltage loop demands
        float vide_error, viqe_error;                   // voltage loop errors
        float kp2, ki2;                                         // voltage loop controller gains
        static float zd2 = 0, zq2 = 0;                  // voltage loop integrator accumulators
        float speed;                                            // Measured shaft speed in RPM
        static float rotor_position_1=0;
        float rotor_position;
        static float speed_e=0;                                         // Measured shaft speed in RPM
        static float theta_r_new=0;
        float speed_e_1=0;                                                         // Measured shaft speed in RPM
        int angle_loop;
        int theta_r_t,theta_r_temp;
        float error_MRAS_temp;
        float division_factor;
  //static int i=0;
        float speed_demand;                                               // Demanded shaft speed in RPM
        float speed_error;
        float Rr_Lr = 0;
        static float pos_demand = 0;
        // Motor Parameters Defined by Yaman Zbede


        float delta_angle;
        float Fai_rr_d_final,Fai_rr_q_final,rotor_position_final;
        static float gg=0;
        float omega_s;
        //static float sum=0;
        ////////////////////////////////////////////
        float pos_error;
        float pos_mech;                                                   // Motor shaft position
        static int memory[Speed_Rep]=0;
        Uint16 ma, mb, mc;                                                // phase modulation index (1st group)
        Uint16 ma1, mb1, mc1;                                    // phase modulation index (2nd group)
        mb1 = 1875;
        mc1 = 1875;

        Uint16 temp1, temp2, temp3, temp4;              // temporary variables - can be useful
        Uint16 theta_m, theta_t;
        float v1 = 0, v2 = 0, v3 = 0;                            // Isolation amplifier inputs
        float vid = 0, viq = 0;                                           // Isolation amp s-frame voltages
        float vide = 0, viqe = 0;                                // Isolation amp e-frame voltages
        float k = 0;
        long int C1, C2, C3, C4, C5, C6, C7;// Frequency profile - horizonal parameters
        float slope;                                                      // Frequency profile - rate of change
        float md0;                                                                 // Frequency profile - mod index
        int loop_count=0;

        CpuTimer0Regs.TCR.bit.TRB = 1;                  // reload timer0 to initiate code timing

        GpioDataRegs.GPACLEAR.bit.GPIO12 = 1; // Pulse LDAC/ low on DAC to latch data into DAC
        GpioDataRegs.GPASET.bit.GPIO12 = 1;

        ISR_count1++;                                                              // count ISR events since start
        //temp_count++;                                                            // relative count (can be useful
for timing things)
        //flag2 = 1;                                                                         // set flag, can be
useful

#if 1   // Count and display CBC trip events
        if (EPwm1Regs.TZFLG.bit.CBC) {
                trip_count++;
                EALLOW;
                EPwm1Regs.TZCLR.bit.CBC = 1;
                EDIS;
        }
#endif

#if 1
        // Get encoder position and encoder pulse width
        pos_count = (unsigned int) EQep1Regs.QPOSCNT;   // capture encoder position

        //////////////////Yaman Zbede/////////////////////////////////////////

        // to represent the encoder position as an electrical angle between 0 and TABLEN
        Position = pos_count;
        Position = Position * 0.17580; //TABLEN_TABLEN
        Position = (int) Position;
        if (Position > TABLEN)
                Position = Position - TABLEN;
        if (Position < 0)
                Position = Position + TABLEN;

        //////////////////////////////////////////////////////////////////////////

        QCTMR_count = (unsigned int) EQep1Regs.QCPRD; // read QCTMR (encoder pulse width)

        // Calculate speed in RPM based on encoder pulse width (exec time = 1.8us)
        if (QCTMR_count != 0)                                   // avoid divide by zero condition
                        {
                if (EQep1Regs.QEPSTS.bit.QDF)                   // test direction bit
                        speed = (float) VELOCITY_SCALE / QCTMR_count;  //(exec time = 1.6us)
                else
                        speed = -(float) VELOCITY_SCALE / QCTMR_count;
        }
        if (QCTMR_count == 0) {
                speed = 0;
        }
        N_RPM = speed;                          // copy value to N-RPM for transfer to LabVIEW
#endif

// This Part of the code is written by Yaman Zbede to find the shaft position depending on speed signal.
/////////////////////Yaman Zbede///////////////////////////////////////

        //Speed_Control Shold be activated when Speed Control is Deactivated
#if 0
        // Create angle theta_e for testing purposes (useful for producing modulation signals)
        // Calibration based on 20kHz ISR repetition rate
        // theta_e ramps between 0 and 720 and returns to 0
        //      theta_e2 = theta_e2 + 72;  // 5ms period
        //      theta_e2 = theta_e2 + 45;  // 80ms period, 25% rated frequency Change theta_e2 limit to 72000
        theta_e2 = theta_e2 + 18;// 20ms period (useful for generating 50Hz waveforms)
        //      theta_e2 = theta_e2 + 36;  // 10ms period
        //      theta_e2 = theta_e2 + 0.012*par2;   // LabVIEW slider controlled par2=3000 => 40ms period
        //      theta_e2 = theta_e2 + 0.006*par2;   // LabVIEW slider controlled par2=3000 => 20ms period

// code added by Yaman to check constant voltage frequency operation
#if 0
        if(ISR_count1%600000<300000)
        par2=3000;
        else
        par2=1500;
#endif

#if 0 //should be one for  80 ms case
        if(theta_e2 >= 72000)
        theta_e2 = 0;

        theta_e = (int)(0.01*theta_e2);
#endif

#if 1 //Yaman_should be one for cases diffrent from 80 ms
        //theta_e2 = theta_e2 + 0.006*3000;             // LabVIEW slider controlled par2=3000 => 20ms period
        if(theta_e2 >= 7200)
        theta_e2 = 0;
        theta_e = (int)(0.1*theta_e2);
#endif

#endif

#if 1
        // Acquire sensor input data and apply gain and offset adjustments
        // The first 6 sensor inputs have out-of-range trip circuits
        ia = (float) -1 * SENSOR_GAIN0
                        * ((int) (AdcRegs.ADCRESULT0 >> 4) - SENSOR_OFFSET0);//      ADCINA0
        connector J40
        ib = (float) -1 * SENSOR_GAIN1
                        * ((int) (AdcRegs.ADCRESULT1 >> 4) - SENSOR_OFFSET1);//      ADCINB0
        connector J44
        ic = (float) -1 * SENSOR_GAIN2
                        * ((int) (AdcRegs.ADCRESULT2 >> 4) - SENSOR_OFFSET2);//      ADCINA1
        connector J48
        i4 = (float) SENSOR_GAIN3
                        * ((int) (AdcRegs.ADCRESULT3 >> 4) - SENSOR_OFFSET3);//      ADCINB1
        connector J52
        i5 = (float) SENSOR_GAIN4
                        * ((int) (AdcRegs.ADCRESULT4 >> 4) - SENSOR_OFFSET4);//      ADCINA2
        connector J56
        i6 = (float) SENSOR_GAIN5
                        * ((int) (AdcRegs.ADCRESULT5 >> 4) - SENSOR_OFFSET5);//      ADCINB2
        connector J60
#endif

//disp1=1000*ia;

#if     1
        // Acquire raw sensor data
        B0 = AdcRegs.ADCRESULT0 >> 4;       //      ADCINA0      connector J40
        B1 = AdcRegs.ADCRESULT1 >> 4;       //      ADCINB0      connector J44
        B2 = AdcRegs.ADCRESULT2 >> 4;       //      ADCINA1      connector J48
        B3 = AdcRegs.ADCRESULT3 >> 4;       //      ADCINB1      connector J52
        B4 = AdcRegs.ADCRESULT4 >> 4;       //      ADCINA2      connector J56
        B5 = AdcRegs.ADCRESULT5 >> 4;       //      ADCINB2      connector J60
#endif

#if 1
        B6 = AdcRegs.ADCRESULT6 >> 4;       //  ADCINA3 connector J64
        B7 = AdcRegs.ADCRESULT7 >> 4;       //  ADCINB3 connector J68
        B8 = AdcRegs.ADCRESULT8 >> 4;       //  ADCINA4 connector J72
        B9 = AdcRegs.ADCRESULT9 >> 4;       //  ADCINB4 connector J78

        B10 = AdcRegs.ADCRESULT10 >> 4;             //  ADCINA5 connector J9-V1
        B11 = AdcRegs.ADCRESULT11 >> 4;             //  ADCINB5 connector J9-V2
        B12 = AdcRegs.ADCRESULT12 >> 4;             //  ADCINA6 connector J9-V3
        B13 = AdcRegs.ADCRESULT13 >> 4;             //  ADCINB6 connector J9-V4
        B14 = AdcRegs.ADCRESULT14 >> 4;             //  ADCINA7 connector J9-V5
        B15 = AdcRegs.ADCRESULT15 >> 4;             //  ADCINB7 connector J9-V6
#endif
        float V_DC=(B6 - 2076)*0.356118;

#if 1
        // 3/2 transform:
        ids = 0.666666 * ia - 0.333333 * ib - 0.333333 * ic;
        iqs = 0.577350 * (ib - ic);



        // 3/2 transform:
        vid = 0.666666 * v1 - 0.333333 * v2 - 0.333333 * v3;
        viq = 0.577350 * (v2 - v3);

        // Reference frame transform:
        ide = ids * cos_t[theta_e] + iqs * sin_t[theta_e];
        iqe = -ids * sin_t[theta_e] + iqs * cos_t[theta_e];

        // Reference frame transform:
        vide = vid * cos_t[theta_e] + viq * sin_t[theta_e];
        viqe = -vid * sin_t[theta_e] + viq * cos_t[theta_e];
```

```
	// Calculate magnitude of stator current vector
	imod = (int) sqrt(ide * ide + iqe * iqe);			//(exec time = 620ns)

#endif

#if 1
	// Current controller code written by Yaman Zbede

	kp = 1474.56;	//Kcr=32768	Pcr=20msec
	ki = 4.42368;

	//speed_control Yaman Zbede Speed Controller
#if 1
	kps = 0.05;
	kis = 0.00003;

	speed_demand = par3;

	//This part of the code is written by Yaman Zbede to reverse the speed slowly by using speed reversal
button
#if 0
	change=reverse*par3*0.00005+change;
	if(par3>0)
	{
		if(change>0)
		change=0;
		if(change<-2*par3)
		change=-2*par3;
	}
	if(par3<=0)
	{
		if(change<=0)
		change=0;
		if(change>-2*par3)
		change=-2*par3;
	}
	speed_demand=par3+change;

#endif
#if 0 //Code Written by Yaman Zbede to Change the speed reference periodically and from positive to nigative
	Speed_count=ISR_count1%480000;
	if((Speed_count<60000))
	par3=200;
	else if((Speed_count<120000))
	par3=500;
	else if((Speed_count<180000))
	par3=1000;
	else if((Speed_count<240000))
	par3=500;
	else if((Speed_count<300000))
	par3=-500;
	else if((Speed_count<360000))
	par3=-1000;
	else if((Speed_count<420000))
	par3=-500;
	else
	par3=-200;
#endif

	// This Part of the code is written by Yaman Zbede to make speed change slowly when slow change
button is pressed
#if 0	//sss change speed from +speed to -speed during 3 seconds
	if(store_enable)
	{
		flag2=1;
		start_time++;
		if(start_time>10000)	// 1sec
```

```
                        par3=abs(par3);
            }
            else
            {
                  flag2=0;
                  start_time=0;

            }


#endif

#if 0   //rrr change speed from +100 to -100 during 50 seconds
            if(store_enable)
            {
                  flag2=1;
                  start_time++;
                  par3=100;
                  if(start_time>45454)   // 1sec
                        par3=75;
                  if(start_time>90909)   // 1sec
                        par3=50;
                  if(start_time>136363)   // 1sec
                        par3=25;
                  if(start_time>181818)   // 1sec
                        par3=0;
                  if(start_time>227272)   // 1sec
                        par3=-25;
                  if(start_time>272727)   // 1sec
                        par3=-50;
                  if(start_time>318181)   // 1sec
                        par3=-75;
                  if(start_time>363636)   // 1sec
                        par3=-100;
            }
            else
            {
                  flag2=0;
                  start_time=0;

            }


#endif
#if 0   //ttt change speed from 25 to 0 to 100 during 50 seconds
            if(store_enable)
            {
                  flag2=1;
                  start_time++;
                  par3=25;
                  if(start_time>45454)   // 1sec
                        par3=0;
                  if(start_time>363636)   // 1sec
                        par3=100;
            }
            else
            {
                  flag2=0;
                  start_time=0;

            }


#endif
#if 0   //uuu change speed from 20 to 10 to 0 during 50 seconds
            if(store_enable)
            {
                flag2=1;
                start_time++;
                par3=20;
                if(start_time>100000)   // 1sec
                        par3=10;
                if(start_time>200000)   // 1sec
                        par3=0;
        }
        else
        {
                flag2=0;
                start_time=0;

        }


#endif

#if 0   //vvv change speed from 20 to -20 during 20 seconds 100DTY
        if(store_enable)
        {
                flag2=1;
                start_time++;
                par3=20;
                if(start_time>100000)   // 1sec
                        par3=-20;

        }
        else
        {
                flag2=0;
                start_time=0;

        }


#endif

#if 1
        if ((reverse == -1) && (par3 >= 0) && (release)) {
                change = 0.02 + change;
                if (change > par3)
                        change = change - 0.04;
                speed_demand = (int) (change + 0.6);
        } else if ((reverse == -1) && (par3 < 0) && (release)) {
                change = -0.02 + change;
                if (change < par3)
                        change = change + 0.04;ipiipi
                speed_demand = (int) (change - 0.6);
        } else if ((reverse == 1) && (release)) {
                change = par3;
                speed_demand = (int) (change);
        } else {
                change = 0;
                speed_demand = (int) (change);
        }

#endif
// Yaman Zbede Sensored vector control
        if (!Relay4_button_status)
                speed_error = (speed_demand - speed);


// Yaman Zbede Sensorless vector control
        if (Relay4_button_status)
                speed_error = (speed_demand - ave1);
```

```
	iqe_demand = kps * speed_error + zs;
	if (release)
		zs = zs + kis * speed_error;

	// d-axis integrator clamp (+/- 95% mod index range)
	if (zs > SPEED_LOOP_INT_CLAMP)
		zs = SPEED_LOOP_INT_CLAMP;
	if (zs < -SPEED_LOOP_INT_CLAMP)
		zs = -SPEED_LOOP_INT_CLAMP;
	//iqe_demand=iqe_demand*0.0001;

	if (iqe_demand > iqe_demand_CLAMP)
		iqe_demand = iqe_demand_CLAMP;
	if (iqe_demand < -iqe_demand_CLAMP)
		iqe_demand = -iqe_demand_CLAMP;

	ide_demand = 1 / Lm;	// Ampere

	// Yaman Zbede This code should be executed when sensored vector control is applied
	if (!Relay4_button_status)
		{
			N_RPM1 = 34.6908042* iqe_demand / ide_demand;  //Slip Speed Calculation
			Pos_Yam = N_RPM1 * 0.0012 + Pos_Yam_1; //converting to electrical angle vary
between 0 and 720 720/60*50e-6
			omega_s=2*speed+N_RPM1;
			omega_s=abs(omega_s);

			if ((Pos_Yam >= 720))
				Pos_Yam = Pos_Yam - 720;
			if ((Pos_Yam <= 0))
				Pos_Yam = Pos_Yam + 720;
			Pos_Yam_1 = Pos_Yam;
			Pos_Yam = Pos_Yam + Position; // here the rotor position is added to the slip position
			if ((Pos_Yam >= 720))
				Pos_Yam = Pos_Yam - 720;
			if ((Pos_Yam <= 0))
				Pos_Yam = Pos_Yam + 720;
		}


		// Yaman Zbede This code should be executed when sensorless vector control is applied
		if (Relay4_button_status)
		{
			N_RPM1 = 34.6908042 * iqe_demand / ide_demand;  //Slip Speed Calculation
			Pos_Yam = N_RPM1 * 0.0012 + Pos_Yam_1; //converting to electrical angle vary
between 0 and 720 720/60*50e-6
			omega_s=2*ave1+N_RPM1;
			omega_s=abs(omega_s);
			if ((Pos_Yam >= 720))
				Pos_Yam = Pos_Yam - 720;
			if ((Pos_Yam <= 0))
				Pos_Yam = Pos_Yam + 720;
			Pos_Yam_1 = Pos_Yam;
			Pos_Yam = Pos_Yam + theta_r; // here the rotor position is added to the slip position
			if ((Pos_Yam >= 720))
				Pos_Yam = Pos_Yam - 720;
			if ((Pos_Yam <= 0))
				Pos_Yam = Pos_Yam + 720;
		}



	theta_e = (int) (Pos_Yam);

#endif
```

```
#if 0  // Speed_control it should be activated when current control is applied
        ide_demand=par3*0.004;
        iqe_demand=ide_demand;

#endif

        id_error = (ide_demand - ide);
        iq_error = (iqe_demand - iqe);

        // d-axis PI with feedforward
        vde = kp * id_error + zd;
        if (release)
                zd = zd + ki * id_error;

        // d-axis integrator clamp (+/- 95% mod index range)
        if (zd > CURRENT_LOOP_INT_CLAMP)
                zd = CURRENT_LOOP_INT_CLAMP;
        if (zd < -CURRENT_LOOP_INT_CLAMP)
                zd = -CURRENT_LOOP_INT_CLAMP;

        // q-axis PI with feedforward
        vqe = kp * iq_error + zq;
        if (release)
                zq = zq + ki * iq_error;

        //  q-axis integrator clamp (+/- 95% mod index range)
        if (zq > CURRENT_LOOP_INT_CLAMP)
                zq = CURRENT_LOOP_INT_CLAMP;
        if (zq < -CURRENT_LOOP_INT_CLAMP)
                zq = -CURRENT_LOOP_INT_CLAMP;

        vde_m = (int) vde;
        vqe_m = (int) vqe;
        ide_m = (int) ide;
        iqe_m = (int) iqe;
#endif

        //  Reference frame transformation (exec time = 150ns)
        vds = vde * cos_t[theta_e] - vqe * sin_t[theta_e];
        vqs = vde * sin_t[theta_e] + vqe * cos_t[theta_e];

#if 1
        va = vds;
        vb = -0.5 * vds + 0.866025 * vqs;
        vc = -0.5 * vds - 0.866025 * vqs;
#endif
#if 1  //*************************deadband compensation***********************************************
        int deadband=69;
        float on_drop=par6;
        if (speed_demand<0)
                on_drop=0.25*par6;
        if((par3!=speed_demand)||(abs(speed_demand)<5)||!release)
                on_drop=0;
        //on_drop=0.1*on_drop;
        float on_d=on_drop*abs(iqe)/speed_demand;
        if((speed_demand)!=par3)
                on_drop=0;
        //if(speed_demand<0)
                //on_d=-1*on_d;
        vde=vde-on_d;
        //if(iqe_demand<0)
                //on_d=-1*on_d;
        vqe=vqe-on_d;
        if(ia>0)
                va+=deadband;
        else
                va-=deadband;
        if(ib>0)
                vb+=deadband;
        else
                vb-=deadband;
        if(ic>0)
                vc+=deadband;
        else
                vc-=deadband;


        //if (par3>0)

        //if (par3<0)
                //vde=vde+on_d;
        vds = vde * cos_t[theta_e] - vqe * sin_t[theta_e];
        vqs = vde * sin_t[theta_e] + vqe * cos_t[theta_e];

#endif//*******************************************************************************************
// Yaman Zbede MRAS
#if 1
        float alpha=par2;
        alpha=alpha*0.01;
        alpha+=1;
        Sai_r_d = 0.078145 * vds - 1.8299 * ids*alpha - 1306.945 * (ids - ids_1);
        Sai_r_q = 0.078145 * vqs - 1.8299 * iqs*alpha - 1306.945 * (iqs - iqs_1);

        //Sai_r_d=Sai_r_d/alpha;
        //Sai_r_q=Sai_r_q/alpha;



#if 0 //integration
        if( Sai_r_d>Sai_r_d_1)
                Sai_r_d_2=Sai_r_d;
        if( Sai_r_d<Sai_r_d_1)
                Sai_r_d_3=Sai_r_d;
        alpha=(Sai_r_d_2+Sai_r_d_3)*0.5;
        Sai_r_d+=alpha;
        if( Sai_r_q>Sai_r_q_1)
                Sai_r_q_2=Sai_r_q;
        if( Sai_r_q<Sai_r_q_1)
                Sai_r_q_3=Sai_r_q;
        alpha=(Sai_r_q_2+Sai_r_q_3)*0.5;
        Sai_r_q+=alpha;

        Sai_r_d = Sai_r_d * 0.0001 + Sai_r_d_1;
        Sai_r_q = Sai_r_q * 0.0001 + Sai_r_d_1;
        Sai_r_d_1=Sai_r_d;
        Sai_r_q_1=Sai_r_q;


#endif

#if 0  //three stage low pass filtering
        if (omega_s<6)
        omega_s=6;
        float tau_s=1/omega_s;
        //float tau=9.549296586*tau_s;  //tan(90/2)/w
        //float tau=5.5132889554*tau_s;  //tan(90/3)/w
        float tau=3.955448157*tau_s;  //tan(90/4)/w
        //float tau=3.102754546*tau_s;  //60/2/pi*tan(90/5)/w
```

```
            float F_a=tau/(0.0001+tau);  //tau/(Ts+tau)
            float F_b=0.00005/(0.0001+tau);  //Ts/(Ts+tau)
            Sai_r_d = 10*Sai_r_d * F_b + F_a * Sai_r_d_1;
            Sai_r_q =10* Sai_r_q * F_b + F_a * Sai_r_q_1;
            Sai_r_d_1=Sai_r_d;
            Sai_r_q_1=Sai_r_q;

            Sai_r_d = Sai_r_d * F_b + F_a * Sai_r_d_2;
            Sai_r_q = Sai_r_q * F_b + F_a * Sai_r_q_2;
            Sai_r_d_2=Sai_r_d;
            Sai_r_q_2=Sai_r_q;

            Sai_r_d = Sai_r_d * F_b + F_a * Sai_r_d_3;
            Sai_r_q = Sai_r_q * F_b + F_a * Sai_r_q_3;
            Sai_r_d_3=Sai_r_d;
            Sai_r_q_3=Sai_r_q;

            Sai_r_d = Sai_r_d * F_b + F_a * Sai_r_d_4;
            Sai_r_q = Sai_r_q * F_b + F_a * Sai_r_q_4;
            Sai_r_d_4=Sai_r_d;
            Sai_r_q_4=Sai_r_q;
/*
            Sai_r_d = Sai_r_d * F_b + F_a * Sai_r_d_5;
            Sai_r_q = Sai_r_q * F_b + F_a * Sai_r_q_5;
            Sai_r_d_5=Sai_r_d;
            Sai_r_q_5=Sai_r_q;
            */
#endif
#if 0

            Sai_r_d = Sai_r_d * 0.000061640 + 0.9994 * Sai_r_d_1;
            Sai_r_q = Sai_r_q * 0.000061640 + 0.9994 * Sai_r_q_1;
            Sai_r_d_1=Sai_r_d;
            Sai_r_q_1=Sai_r_q;
#endif
#if 1

            Sai_r_d = Sai_r_d * 0.0000627924*0.8726 + 0.999372 * Sai_r_d_1+valpha;
            Sai_r_q = Sai_r_q * 0.0000627924*0.8726+ 0.999372* Sai_r_q_1+vbeta;
            Sai_r_d_1=Sai_r_d;
            Sai_r_q_1=Sai_r_q;
#endif

//=========================================PI    replacement    with    itteration    method
IPI========================================
//================================================================================
========================================
      theta_r_new=0;
      theta_r_t=0;
      error_MRAS=0;
     // if ((release)&((ISR_count1 % 1000 == 0)||(ISR_count1 % 999 == 0)||(ISR_count1 % 998 == 0)||(ISR_count1
% 997 == 0)))
      if ((release)&&(ISR_count1 % 1 == 0))
      {
               for (loop_count=0;loop_count<8;loop_count++)
               {
                            if((error_MRAS1<100)&&(loop_count==1))
                            {
                                         loop_count=7;
                                         theta_r_new=error_MRAS2;

                            }
                            //theta_r_new=theta_r_t;
                            //division_factor=2^loop_count;
                            //division_factor=2;
                            //delta_angle=180/division_factor;
            switch (loop_count)
            {
            case 0: delta_angle=128;break;
            case 1: delta_angle=64;break;
            case 2: delta_angle=32;break;
            case 3: delta_angle=16;break;
            case 4: delta_angle=8;break;
            case 5: delta_angle=4;break;
            case 6: delta_angle=2;break;
            case 7: delta_angle=1;break;




            }


            for(angle_loop=0;angle_loop<8;angle_loop++)
            {
                theta_r_temp=(int)(theta_r_new+delta_angle*(angle_loop-4));
                if(theta_r_temp<0)
                    theta_r_temp+=TABLEN;
                if(theta_r_temp>=TABLEN)
                    theta_r_temp=theta_r_temp-TABLEN;
                // Reference frame transform: rotor speed rotating reference frame Yaman Zbede
                idr = ids * cos_t[theta_r_temp] + iqs * sin_t[theta_r_temp];
                iqr = -ids * sin_t[theta_r_temp] + iqs * cos_t[theta_r_temp];
                 //Lm*Ls*Lr*Rs*Rr*sigma;

                Fai_rr_d  =0.000100143*  idr  +0.9997  *  Fai_rr_d_1;//Ts/Tr*Lm  *  idr  -  (Ts/Tr-1)  *
Fai_rr_d_1;
                Fai_rr_q =0.000100143* iqr +0.9997* Fai_rr_q_1;
                //  Reference frame transformation (exec time = 150ns)
                Fai_r_d = Fai_rr_d * cos_t[theta_r_temp] - Fai_rr_q * sin_t[theta_r_temp];
                Fai_r_q = Fai_rr_d* sin_t[theta_r_temp] + Fai_rr_q * cos_t[theta_r_temp];
                error_MRAS_temp = 100*Fai_r_d*Sai_r_q-100*Fai_r_q*Sai_r_d;
                //ave1=error_MRAS_temp;
                /*if((loop_count==0))
                {
                    if(angle_loop==1)
                    {
                        error_MRAS1=error_MRAS_temp;
                    }
                    else if(angle_loop==2)
                        error_MRAS2=error_MRAS_temp;
                    else if(angle_loop==0)
                        error_MRAS0=error_MRAS_temp;
                    else if(angle_loop==3)
                    {
                        error_MRAS3=error_MRAS_temp;
                    }
                }*/
                if((error_MRAS_temp>error_MRAS)&&(error_MRAS_temp>0))
                {
                    error_MRAS=error_MRAS_temp;
                    theta_r_t=theta_r_temp;
                    //
                    //rotor_position_1=rotor_position;
                    ave1=theta_r_temp;
                }
            }
            theta_r_new=theta_r_t;
            //ave1=theta_r;
            //error_MRAS = error_MRAS - error_MRAS_1;
        }
    }
    error_MRAS1=theta_r_t-error_MRAS2;
    error_MRAS2=theta_r_t;
    if (error_MRAS1>500)
          error_MRAS1=error_MRAS1-TABLEN;
    if (error_MRAS1<-500)
      error_MRAS1=error_MRAS1+TABLEN;


    sum-=memory[i];
    memory[i]=error_MRAS1;
          sum+=memory[i];
          i++;
          if(i>=Speed_Rep)
                  i=0;
          //sum=0.01*sum;
          ave1=(int)(speed_div*sum);
          speed_new=speed_div*sum;
          ave2=(int)(100*373.0194 * 0.1 * iqe_demand / ide_demand);
          speed_new_1=speed_new;


          //ave1=speed_e;
          //rotor_position_1=rotor_position_final;
          if (par5==0)
                  theta_r=(int)(Position);
          else
          {
                  theta_r=(int)(theta_r_new+180);
                  if (theta_r>=TABLEN)
                          theta_r-=TABLEN;
                  if (theta_r<0)
                          theta_r+=TABLEN;
          }
          // Reference frame transform: rotor speed rotating reference frame Yaman Zbede
          idr = ids * cos_t[theta_r] + iqs * sin_t[theta_r];
          iqr = -ids * sin_t[theta_r] + iqs * cos_t[theta_r];
          Fai_rr_d =0.000100143* idr +0.9997 * Fai_rr_d_1;//Ts/Tr*Lm * idr - (Ts/Tr-1) * Fai_rr_d_1;
          Fai_rr_q =0.000100143* iqr +0.9997* Fai_rr_q_1;
          Fai_r_d = Fai_rr_d * cos_t[theta_r] - Fai_rr_q * sin_t[theta_r];
          Fai_r_q = Fai_rr_d* sin_t[theta_r] + Fai_rr_q * cos_t[theta_r];
          ids_1 = ids;
          iqs_1 = iqs;
          Fai_rr_d_1 = Fai_rr_d;
          Fai_rr_q_1 = Fai_rr_q;
          error_MRAS=10000*Sai_r_q*Fai_r_d-10000*Sai_r_d*Fai_r_q;
          float a=sqrt(Sai_r_d*Sai_r_d+Sai_r_q*Sai_r_q);
                float b=sqrt(Fai_r_d*Fai_r_d+Fai_r_q*Fai_r_q);
                kp_alpha=par5;
                //kp_alpha-=1;
                kp_alpha=0.001*(kp_alpha);
                ki_alpha=0;
                        ki_alpha=0.001*ki_alpha;

           error_alpha=Fai_r_d-Sai_r_d;
           error_beta=Fai_r_q-Sai_r_q;
           valpha=error_alpha*kp_alpha+zd_alpha;
           zd_alpha=zd_alpha+ki_alpha*error_alpha;
          vbeta=error_beta*kp_alpha+zd_beta;
          zd_beta=zd_beta+ki_alpha*error_beta;


#endif
//svm(-vds,-vqs,&ma,&mb,&mc);                              // Space Vector Modulator (exec time = 1.36us
max)

#if 1
        // add mod index offset (10kHz PWM)
        ma=3750 + va;
        mb=3750 + vb;
        mc=3750 + vc;
#endif

#if 0
        // add mod index offset (20kHz PWM)
        ma = 1875 + va;
        mb = 1875 + vb;
        mc = 1875 + vc;
#endif

#if 1   // Update PWM modulation registers (group 1)
        EPwm1Regs.CMPA.half.CMPA = ma;              // u-phase
        EPwm2Regs.CMPA.half.CMPA = mb;              // v-phase
        EPwm3Regs.CMPA.half.CMPA = mc;              // w-phase
#endif
        // Acode Written By Yaman Zbede To Activate The Brake Chopper
#if 1
        // the voltage sensor output is B6
        //the sensor offset is 2073
        //the sensor gain is 0.3796
        if (B6 >= 2863) //the lower limit           280         210
                chopper = (B6 - 2863) * 28.4041; //144.23 is 3750/(upper limit - lower limit))
        if (B6 < 2863) //the lower limit
                chopper = 0;
        if (chopper > 3750)
                chopper = 3750;
        ma1 = (int) chopper;
#endif

#if 1  // Update PWM modulation registers (group 2)
        EPwm4Regs.CMPA.half.CMPA = ma1;             // u-phase
        EPwm5Regs.CMPA.half.CMPA = mb1;             // v-phase
        EPwm6Regs.CMPA.half.CMPA = mc1;             // w-phase
#endif

        // Send data via SPI to 10-bit DAC channels (limit data range to 0=>1023)

        temp1 = Sai_r_d * 60 + 500;
        temp3 = Fai_r_d * 60 + 500;
        temp2 = Sai_r_q * 60 + 500;
        temp4 = Fai_r_q * 60 + 500;
        if ((Relay1_button_status)&&(Relay2_button_status))
        {
                temp1 = Sai_r_d * 60 + 500;
                temp2 = Fai_r_d * 60 + 500;
                temp3 = Sai_r_q * 60 + 500;
                temp4 = Fai_r_q * 60 + 500;
        }

        disp1 = N_RPM;
        // Due to unresolved issue with TLV5604 DAC, need to send data twice
        SpiaRegs.SPITXBUF = 0x1000 | (temp1 << 2);
        SpiaRegs.SPITXBUF = 0x1000 | (temp1 << 2);
        SpiaRegs.SPITXBUF = 0x5000 | (temp2 << 2);
        SpiaRegs.SPITXBUF = 0x5000 | (temp2 << 2);
        //SpiaRegs.SPITXBUF = 0x9000 | (temp3 << 2);
        SpiaRegs.SPITXBUF = 0x9000 | (temp3 << 2);
        SpiaRegs.SPITXBUF = 0xD000 | (temp4 << 2);
        SpiaRegs.SPITXBUF = 0xD000 | (temp4 << 2);

        // Send selected data to external RAM store
#if 0
        if((speed_demand-speed)>10)
        {
                flag2 = 1;          // extra store enable qualifier (can be useful)
        }
```

```
	if(!store_enable)
		flag2 = 0;
#endif
	flag2=1;
//=====================================================DTY============================
=================================
	if (store_enable && flag2 && (ISR_count1 % 50 == 0)
				&& (store_counter < DATA_STORE_LEN)) {
		s1[store_counter] =  (int)(speed_demand);
				//s2[store_counter] = (int) (100*sin_t[theta_e]);
				//s1[store_counter] = (int) disp1;
				//s2[store_counter] = (int) disp2;
				s2[store_counter] = (int)(speed);
				s3[store_counter] = (int)  (100*speed_new);
				s4[store_counter] = (int) (error_MRAS);
		s5[store_counter] = (int) (Position);		// (1000*Fai_r_q);
		//s4[store_counter] = (int) vde;
		//s5[store_counter] = (int) vqe;
		//s6[store_counter] = (int) 1000*ia;
		//s7[store_counter] = (int) 1000*ib;
		s6[store_counter] = (int) (theta_r);
		s7[store_counter] = (int) (theta_r);		// (vb);
		s8[store_counter] = (int) (speed_e);		// 1000*iqe;
		//s6[store_counter] = (int)B0;
		//s7[store_counter] = (int)B1;
		//s8[store_counter] = (int)B2;
		store_counter++;			//increment store counter
		acc1 += v1;					//sum selected variables
		acc2 += v2;
		acc3 += v3;
	}
	// The following code can be useful for calculating averages
#if 0
	ave1 = (int)(0.000488281*acc1);
	ave2 = (int)(0.000488281*acc2);
	ave3 = (int)(0.000488281*acc3);
#endif

	// Housekeeping at end of ISR
	AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;		// Reset SEQ1
	AdcRegs.ADCTRL2.bit.RST_SEQ2 = 1;		// Reset SEQ2
	AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;		// Clear INT SEQ1 bit
	AdcRegs.ADCST.bit.INT_SEQ2_CLR = 1;		// Clear INT SEQ2 bit

	// Acknowledge this interrupt to receive more interrupts from group 1
	PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;

	T0_count = CpuTimer0Regs.TIM.half.LSW;   // Latch lower 16-bits of timer0

}	// End of ISR

//---------------------------------------------------------------------------------------------------
// EQEP1 interrupt (for testing only)
//

interrupt void eqep1_isr(void)

{
	if (EQep1Regs.QFLG.bit.WTO == 1)
		enc_wdog++;

	if (EQep1Regs.QFLG.bit.QDC == 1)
		reversal_count++;

//	EQep1Regs.QCLR.all = 0x9;			// Clear QDC and INT flags
	EQep1Regs.QCLR.all = 0x19;			// Clear WTO, QDC and INT flags
```

```
		PieCtrlRegs.PIEACK.all = PIEACK_GROUP5;
}
//-----------------------------------------------------------------------------------------------------
// Set up and initialise data stores in external RAM (ZONE7C)
// Each integer array is 2k long as defined by DATA_STORE_LEN
// The arrays are global
#pragma CODE_SECTION(set_up_data_stores, "codeA")
void set_up_data_stores(void) {
		Uint16 i;

		s1 = (int *) calloc(DATA_STORE_LEN, sizeof(int));
		s2 = (int *) calloc(DATA_STORE_LEN, sizeof(int));
		s3 = (int *) calloc(DATA_STORE_LEN, sizeof(int));
		s4 = (int *) calloc(DATA_STORE_LEN, sizeof(int));
		s5 = (int *) calloc(DATA_STORE_LEN, sizeof(int));
		s6 = (int *) calloc(DATA_STORE_LEN, sizeof(int));
		s7 = (int *) calloc(DATA_STORE_LEN, sizeof(int));
		s8 = (int *) calloc(DATA_STORE_LEN, sizeof(int));

		// zero out store - note that this is not necessary when arrays are defined with calloc()
		// can be useful if if initialising with other data is desired
		for (i = 0; i < DATA_STORE_LEN; ++i) {
				s1[i] = 0x0;
				s2[i] = 0x0;
				s3[i] = 0x0;
				s4[i] = 0x0;
				s5[i] = 0x0;
				s6[i] = 0x0;
				s7[i] = 0x0;
				s8[i] = 0x0;
		}
}
//-----------------------------------------------------------------------------------------------------
// Set up floating point sine and cosine tables in external RAM
// Use standard C trig functions sin and cos to build lookup tables
// These are floating point tables with max value of 1.0
#pragma CODE_SECTION(set_up_look_up_tables, "codeA")
void set_up_look_up_tables(void) {
		Uint16 i;
		float angle;

		for (i = 0; i < TABLEN; ++i) {
				angle = 2 * PI * i / TABLEN;
				sin_t[i] = 1.0 * sin(angle);
				cos_t[i] = 1.0 * cos(angle);
		}
}
//-----------------------------------------------------------------------------------------------------
// Control panel control
// Inspect LabVIEW UCP and execute button actions
#pragma CODE_SECTION(panel_controls, "codeA")
void panel_controls(void) {
// Clear PWM trip
		if (Clear_PWM_trip_button_status) {
				Clear_PWM_trip();
				release = 1;
		} else
				pb1 = 0;

		if (Force_PWM_trip_button_status) {
				Force_PWM_trip();
				release = 0;
		} else
				pb7 = 0;

// switch relays based on UCP buttons
		Relay1(Relay1_button_status);
		Relay2(Relay2_button_status);
		Relay3(Relay3_button_status);
		Relay4(Relay4_button_status);

// Gate data store button status
		if (Data_Store_button_status) {
				store_enable = 1;
//				pb3 = 400;
		} else {
				acc1 = 0;
				acc2 = 0;
				acc3 = 0;
				store_enable = 0;
				store_counter = 0;
//				pb3 = 401;
		}

// Comp 2 button
		if (Comp_2_button_status) {
				reverse = -1;
		} else {
				reverse = 1;
		}

}
//--------------------------------------------------------------------------
// Send Monitoring Parameter Frame (MPF) to LabVIEW
// The MPF consists of 15 selected variables to be displayed on LabVIEW UCP
// The size of the MPF can be increased by modification to sprintf statements
// Corresponding changes will be needed to the LabVIEW panel application
// Multiple sprintf() functions are used to limit length of sprintf argument list
void update_panel(void) {
		Uint16 nc;							// sprintf error code
		static char sbuf[100];				// output string for sprintf

		//nc = sprintf(sbuf,"%u %u %u %d \r\0",B0,B1,B2,Ia);	//14-03-2014 phase a current displayed on the 4th 
display
		nc = sprintf(sbuf, "%u %u %u %d \r\0", B0, B1, B2, disp1);
		scia_msg(sbuf);
		nc = sprintf(sbuf, "%lu %lu %u %u \r\0", LoopCount, ISR_count1, pos_count,
						QCTMR_count);
		scia_msg(sbuf);
		nc = sprintf(sbuf, "%d %u %d %d %u %d %d\r\n\0", N_RPM, reversal_count,
						imod, (B6 - 2076), T0_count, ave1, TZflag);
		scia_msg(sbuf);
}
//--------------------------------------------------------------------------
// Send values to LabVIEW Data Transfer Interface (DTI)
// This function assembles the data from the 8 ext RAM data store contents
// into frames and transfers the data to the LabVIEW DTI. The LabVIEW DTI
// sends a frame identifier (frame_count) and this function responds by
// sending 32 words of 16-bit data back to the LabVIEW DTI via the RS232 link.
// This process repeats until all the requested frames are sent.
// More details of the data transfer process and frame structure are
// given in the Software Design Report.
//
void transfer_store_data_to_RS232(int frame_count) {
		int d01, d02, d03, d04, d05, d06, d07, d08, d09, d10, d11, d12, d13, d14,
						d15, d16;
		int d17, d18, d19, d20, d21, d22, d23, d24, d25, d26, d27, d28, d29, d30,
						d31, d32;
		Uint16 index1, index2, index3, index4;
		Uint16 nc;
		static char sbuf[100];							// output string for sprintf

		index1 = 4 * frame_count;
		index2 = index1 + 1;
```

```
        index3 = index2 + 1;
        index4 = index3 + 1;

        d01 = s1[index1];
        d02 = s2[index1];
        d03 = s3[index1];
        d04 = s4[index1];
        d05 = s5[index1];
        d06 = s6[index1];
        d07 = s7[index1];
        d08 = s8[index1];

        d09 = s1[index2];
        d10 = s2[index2];
        d11 = s3[index2];
        d12 = s4[index2];
        d13 = s5[index2];
        d14 = s6[index2];
        d15 = s7[index2];
        d16 = s8[index2];

        d17 = s1[index3];
        d18 = s2[index3];
        d19 = s3[index3];
        d20 = s4[index3];
        d21 = s5[index3];
        d22 = s6[index3];
        d23 = s7[index3];
        d24 = s8[index3];

        d25 = s1[index4];
        d26 = s2[index4];
        d27 = s3[index4];
        d28 = s4[index4];
        d29 = s5[index4];
        d30 = s6[index4];
        d31 = s7[index4];
        d32 = s8[index4];

        nc = sprintf(sbuf, "%d %d %d %d %d %d %d %d %d \r\0", index1, d01, d02, d03,
                        d04, d05, d06, d07, d08);
        scia_msg(sbuf);
        nc = sprintf(sbuf, "%d %d %d %d %d %d %d %d %d \r\0", index2, d09, d10, d11,
                        d12, d13, d14, d15, d16);
        scia_msg(sbuf);
        nc = sprintf(sbuf, "%d %d %d %d %d %d %d %d %d \r\0", index3, d17, d18, d19,
                        d20, d21, d22, d23, d24);
        scia_msg(sbuf);
        nc = sprintf(sbuf, "%d %d %d %d %d %d %d %d %d \r\n\0", index4, d25, d26,
                        d27, d28, d29, d30, d31, d32);
        scia_msg(sbuf);
}
//===========================================================================
// End of code
//===========================================================================
```

REFERENCES

[1] J. W. Finch and D. Giaouris, "Controlled AC electrical drives," *Ieee Transactions on Industrial Electronics,* vol. 55, pp. 481-491, Feb 2008.

[2] P. Vas, *Sensorless vector and direct torque control*: Oxford University Press, USA, 1998.

[3] J. Holtz, "Sensorless control of induction motor drives," *Proceedings of the Ieee,* vol. 90, pp. 1359-1394, Aug 2002.

[4] J. R. Heredia, F. P. Hidalgo, and J. L. D. Paz, "Sensorless control of induction motors by artificial neural networks," *Ieee Transactions on Industrial Electronics,* vol. 48, pp. 1038-1040, Oct 2001.

[5] K. D. Hurst, T. G. Habetler, G. Griva, and F. Profumo, "Zero-speed tacholess IM torque control: simply a matter of stator voltage integration," *IEEE Transactions on industry applications,* vol. 34, pp. 790-795, 1998.

[6] J. Holtz and J. T. Quan, "Drift- and parameter-compensated flux estimator for persistent zero-stator-frequency operation of sensorless-controlled induction motors," *Ieee Transactions on Industry Applications,* vol. 39, pp. 1052-1060, Jul-Aug 2003.

[7] M. Rashed and A. F. Stronach, "A stable back-EMF MRAS-based sensorless low-speed induction motor drive insensitive to stator resistance variation," *Iee Proceedings-Electric Power Applications,* vol. 151, pp. 685-693, Nov 2004.

[8] J. Holtz, "Sensorless control of induction machines - With or without signal injection?," *Ieee Transactions on Industrial Electronics,* vol. 53, pp. 7-30, Feb 2006.

[9] H. U. Rehman, A. Derdiyok, M. K. Guven, and L. Y. Xu, "A new current model flux observer for wide speed range sensorless control of an induction machine," *Ieee Transactions on Power Electronics,* vol. 17, pp. 1041-1048, Nov 2002.

[10] P. Cortes, M. P. Kazmierkowski, R. M. Kennel, D. E. Quevedo, and J. Rodriguez, "Predictive Control in Power Electronics and Drives," *Ieee Transactions on Industrial Electronics,* vol. 55, pp. 4312-4324, Dec 2008.

[11] H. Miranda, P. Cortes, J. I. Yuz, and J. Rodriguez, "Predictive Torque Control of Induction Machines Based on State-Space Models," *Ieee Transactions on Industrial Electronics,* vol. 56, pp. 1916-1924, Jun 2009.

[12] J. Rodriguez, R. M. Kennel, J. R. Espinoza, M. Trincado, C. A. Silva, and C. A. Rojas, "High-Performance Control Strategies for Electrical Drives: An Experimental Assessment," *Ieee Transactions on Industrial Electronics,* vol. 59, pp. 812-820, Feb 2012.

[13] T. Geyer, G. Papafotiou, and M. Morari, "Model Predictive Direct Torque Control-Part I: Concept, Algorithm, and Analysis," *Ieee Transactions on Industrial Electronics,* vol. 56, pp. 1894-1905, Jun 2009.

[14] G. Papafotiou, J. Kley, K. G. Papadopoulos, P. Bohren, and M. Morari, "Model Predictive Direct Torque Control-Part II: Implementation and Experimental Evaluation," *Ieee Transactions on Industrial Electronics,* vol. 56, pp. 1906-1915, Jun 2009.

[15] M. R. Arahal, F. Barrero, S. Toral, M. Duran, and R. Gregor, "Multi-phase current control using finite-state model-predictive control," *Control Engineering Practice,* vol. 17, pp. 579-587, May 2009.

[16] F. Barrero, M. R. Arahal, R. Gregor, S. Toral, and M. J. Duran, "A Proof of Concept Study of Predictive Current Control for VSI-Driven Asymmetrical Dual Three-Phase

AC Machines," *Ieee Transactions on Industrial Electronics,* vol. 56, pp. 1937-1954, Jun 2009.

[17] R. Morales-Caporal, E. Bonilla-Huerta, C. Hernandez, M. A. Arjona, and M. Pacas, "Transducerless Acquisition of the Rotor Position for Predictive Torque Controlled PM Synchronous Machines Based on a DSP-FPGA Digital System," *Ieee Transactions on Industrial Informatics,* vol. 9, pp. 799-807, May 2013.

[18] L. Tong, X. D. Zou, S. S. Feng, Y. Chen, Y. Kang, Q. J. Huang*, et al.*, "An SRF-PLL-Based Sensorless Vector Control Using the Predictive Deadbeat Algorithm for the Direct-Driven Permanent Magnet Synchronous Generator," *Ieee Transactions on Power Electronics,* vol. 29, pp. 2837-2849, Jun 2014.

[19] R. Morales-Caporal, E. Bonilla-Huerta, M. A. Arjona, and C. Hernandez, "Sensorless Predictive DTC of a Surface-Mounted Permanent-Magnet Synchronous Machine Based on Its Magnetic Anisotropy," *Ieee Transactions on Industrial Electronics,* vol. 60, pp. 3016-3024, Aug 2013.

[20] J. L. Chen and T. H. Liu, "Implementation of a predictive controller for a sensorless interior permanent-magnet synchronous motor drive system," *Iet Electric Power Applications,* vol. 6, pp. 513-525, Sep 2012.

[21] F. Wang, S. A. Davari, Z. Chen, Z. Zhang, D. A. Khaburi, J. Rodriguez*, et al.*, "Finite Control Set Model Predictive Torque Control of Induction Machine with a Robust Adaptive Observer," *IEEE Transactions on Industrial Electronics,* vol. 64, pp. 2631-2641, 2017.

[22] F. R. Salmasi and T. A. Najafabadi, "An Adaptive Observer With Online Rotor and Stator Resistance Estimation for Induction Motors With One Phase Current Sensor," *Ieee Transactions on Energy Conversion,* vol. 26, pp. 959-966, Sep 2011.

[23] S. A. Davari, D. A. Khaburi, F. X. Wang, and R. M. Kennel, "Using Full Order and Reduced Order Observers for Robust Sensorless Predictive Torque Control of Induction Motors," *Ieee Transactions on Power Electronics,* vol. 27, pp. 3424-3433, Jul 2012.

[24] J. Guzinski and H. Abu-Rub, "Speed Sensorless Induction Motor Drive With Predictive Current Controller," *Ieee Transactions on Industrial Electronics,* vol. 60, pp. 699-709, Feb 2013.

[25] P. Alkorta, O. Barambones, J. A. Cortajarena, and A. Zubizarrreta, "Efficient Multivariable Generalized Predictive Control for Sensorless Induction Motor Drives," *Ieee Transactions on Industrial Electronics,* vol. 61, pp. 5126-5134, Sep 2014.

[26] F. X. Wang, Z. Chen, P. Stolze, J. F. Stumper, J. Rodriguez, and R. Kennel, "Encoderless Finite-State Predictive Torque Control for Induction Machine With a Compensated MRAS," *Ieee Transactions on Industrial Informatics,* vol. 10, pp. 1097-1106, May 2014.

[27] J. Y. Ruan and S. M. Wang, "A Prediction Error Method-Based Self-Commissioning Scheme for Parameter Identification of Induction Motors in Sensorless Drives," *Ieee Transactions on Energy Conversion,* vol. 30, pp. 384-393, Mar 2015.

[28] J. CatalaiLopez, L. Romeral, A. Arias, and E. Aldabas, "Novel fuzzy adaptive sensorless induction motor drive," *IEEE Transactions on Industrial Electronics,* vol. 53, pp. 1170-1178, 2006.

[29] A. V. R. Teja, V. Verma, and C. Chakraborty, "A New Formulation of Reactive-Power-Based Model Reference Adaptive System for Sensorless Induction Motor Drive," *Ieee Transactions on Industrial Electronics,* vol. 62, pp. 6797-6808, Nov 2015.

[30] F. Alonge, F. D'Ippolito, and A. Sferlazza, "Sensorless Control of Induction-Motor Drive Based on Robust Kalman Filter and Adaptive Speed Estimation," *Ieee Transactions on Industrial Electronics,* vol. 61, pp. 1444-1453, Mar 2014.

[31] M. Habibullah and D. D. C. Lu, "A Speed-Sensorless FS-PTC of Induction Motors Using Extended Kalman Filters," *Ieee Transactions on Industrial Electronics,* vol. 62, pp. 6765-6778, Nov 2015.

[32] S. D. Gennaro, J. R. Dom, x00Ed, nguez, and M. A. Meza, "Sensorless High Order Sliding Mode Control of Induction Motors With Core Loss," *IEEE Transactions on Industrial Electronics,* vol. 61, pp. 2678-2689, 2014.

[33] L. H. Zhao, J. Huang, H. Liu, B. N. Li, and W. B. Kong, "Second-Order Sliding-Mode Observer With Online Parameter Identification for Sensorless Induction Motor Drives," *Ieee Transactions on Industrial Electronics,* vol. 61, pp. 5280-5289, Oct 2014.

[34] S. M. Gadoue, D. Giaouris, and J. W. Finch, "Sensorless Control of Induction Motor Drives at Very Low and Zero Speeds Using Neural Network Flux Observers," *Ieee Transactions on Industrial Electronics,* vol. 56, pp. 3029-3039, Aug 2009.

[35] M. Barut, S. Bogosyan, and M. Gokasan, "Speed-sensorless estimation for induction motors using extended Kalman filters," *Ieee Transactions on Industrial Electronics,* vol. 54, pp. 272-280, Feb 2007.

[36] K. Young-Real, S. Seung-Ki, and P. Min-Ho, "Speed sensorless vector control of induction motor using extended Kalman filter," *IEEE Transactions on Industry Applications,* vol. 30, pp. 1225-1233, 1994.

[37] K. L. Shi, T. F. Chan, Y. K. Wong, and S. L. Ho, "Speed estimation of an induction motor drive using an optimized extended Kalman filter," *Ieee Transactions on Industrial Electronics,* vol. 49, pp. 124-133, Feb 2002.

[38] B. Akin, U. Orguner, A. Ersak, and M. Ehsani, "Simple derivative-free nonlinear state observer for sensorless AC drives," *Ieee-Asme Transactions on Mechatronics,* vol. 11, pp. 634-643, Oct 2006.

[39] S. Buyamin and J. W. Finch, "Tuning extended Kalman filter for induction motor drives using simulated annealing," in *Modelling, Identification and Control Conference, Innsbruck, Austria*, 2007.

[40] O. Barambones, A. J. Garrido, and F. J. Maseda, "Integral sliding-mode controller for induction motor based on field-oriented control theory," *Iet Control Theory and Applications,* vol. 1, pp. 786-794, May 2007.

[41] Z. Yan, C. X. Jin, and V. I. Utkin, "Sensorless sliding-mode control of induction motors," *Ieee Transactions on Industrial Electronics,* vol. 47, pp. 1286-1297, Dec 2000.

[42] S. M. Gadoue, D. Giaouris, and J. W. Finch, "MRAS Sensorless Vector Control of an Induction Motor Using New Sliding-Mode and Fuzzy-Logic Adaptation Mechanisms," *Ieee Transactions on Energy Conversion,* vol. 25, pp. 394-402, Jun 2010.

[43] M. Comanescu and L. Y. Xu, "Sliding-mode MRAS speed estimators for sensorless vector control of induction machine," *Ieee Transactions on Industrial Electronics,* vol. 53, pp. 146-153, Feb 2006.

[44] S. M. Gadoue, D. Giaouris, and J. W. Finch, "A new fuzzy logic based adaptation mechanism for MRAS sensorless vector control induction motor drives," in *Power Electronics, Machines and Drives, 2008. PEMD 2008. 4th IET Conference on*, 2008, pp. 179-183.

[45] M. Cirrincione and M. Pucci, "An MRAS-based sensorless high-performance induction motor drive with a predictive adaptive model," *Ieee Transactions on Industrial Electronics,* vol. 52, pp. 532-551, Apr 2005.

[46] C. Schauder, "Adaptive Speed Identification for Vector Control of Induction-Motors without Rotational Transducers," *Ieee Transactions on Industry Applications,* vol. 28, pp. 1054-1061, Sep-Oct 1992.

[47] F. Z. Peng and T. Fukao, "Robust Speed Identification for Speed-Sensorless Vector Control of Induction-Motors," *Ieee Transactions on Industry Applications,* vol. 30, pp. 1234-1240, Sep-Oct 1994.

[48] G. J. Armstrong, D. J. Atkinson, and P. P. Acarnley, "A comparison of estimation techniques for sensorless vector controlled induction motor drives," in *Power Electronics and Drive Systems, 1997. Proceedings., 1997 International Conference on*, 1997, pp. 110-116 vol.1.

[49] K. J. Åström and B. Wittenmark, *Adaptive control*: Courier Corporation, 2013.

[50] H. Shoudao, W. Yaonan, G. Jian, L. Jiantao, and Q. Sihai, "The vector control based on MRAS speed sensorless induction motor drive," in *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, 2004, pp. 4550-4553 Vol.5.

[51] Y. D. Landau, "Adaptive control: The model reference approach," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. SMC-14, pp. 169-170, 1984.

[52] H. Rehman, A. Derdiyok, M. K. Guven, and X. Longya, "An MRAS scheme for on-line rotor resistance adaptation of an induction machine," in *Power Electronics Specialists Conference, 2001. PESC. 2001 IEEE 32nd Annual*, 2001, pp. 817-822 vol.2.

[53] Y. Zbede, S. M. Gadoue, D. J. Atkinson, and M. A. Elgendy, "Predictive sensorless control of induction motor drives," in *Industrial Technology (ICIT), 2015 IEEE International Conference on*, 2015, pp. 2339-2344.

[54] P. Fang-Zheng, T. Fukao, and L. Jih-Sheng, "Low-speed performance of robust speed identification using instantaneous reactive power for tacholess vector control of induction motors," in *Industry Applications Society Annual Meeting, 1994., Conference Record of the 1994 IEEE*, 1994, pp. 509-514 vol.1.

[55] S. Maiti, C. Chakraborty, Y. Hori, and M. C. Ta, "Model reference adaptive controller-based rotor resistance and speed estimation techniques for vector controlled induction motor drive utilizing reactive power," *Ieee Transactions on Industrial Electronics,* vol. 55, pp. 594-601, Feb 2008.

[56] T. Cao-Minh, T. Uchida, and Y. Hori, "MRAS-based speed sensorless control for induction motor drives using instantaneous reactive power," in *Industrial Electronics Society, 2001. IECON '01. The 27th Annual Conference of the IEEE*, 2001, pp. 1417-1422 vol.2.

[57] M. S. Nait Said and M. E. H. Benbouzid, "Induction motors direct field oriented control with robust on-line tuning of rotor resistance," *IEEE Transactions on Energy Conversion,* vol. 14, pp. 1038-1042, 1999.

[58] A. N. Smith, S. M. Gadoue, and J. W. Finch, "Improved Rotor Flux Estimation at Low Speeds for Torque MRAS-Based Sensorless Induction Motor Drives," *IEEE Transactions on Energy Conversion,* vol. 31, pp. 270-282, 2016.

[59] C. Schauder, "Adaptive speed identification for vector control of induction motors without rotational transducers," in *Industry Applications Society Annual Meeting, 1989., Conference Record of the 1989 IEEE*, 1989, pp. 493-499 vol.1.

[60] R. Blasco-Giménez, G. M. Asher, M. Sumner, and K. J. Bradley, "Dynamic performance limitations for MRAS based sensorless induction motor drives. Part 2:

Online parameter tuning and dynamic performance studies," *IEE Proceedings - Electric Power Applications,* vol. 143, p. 123, 1996.

[61] W. Zhigan, Z. Dawei, and Y. Jianping, "Research on speed estimation algorithm for induction motor drive," in *Power Electronics and Motion Control Conference, 2004. IPEMC 2004. The 4th International*, 2004, pp. 1387-1392 Vol.3.

[62] M. Tsuji, S. Chen, K. Izumi, and E. Yamada, "A sensorless vector control system for induction motors using q-axis flux with stator resistance identification," *Ieee Transactions on Industrial Electronics,* vol. 48, pp. 185-194, Feb 2001.

[63] R. Blasco-Giménez, G. M. Asher, M. Sumner, and K. J. Bradley, "Dynamic performance limitations for MRAS based sensorless induction motor drives. Part 1: Stability analysis for the closed loop drive," *IEE Proceedings - Electric Power Applications,* vol. 143, p. 113, 1996.

[64] K. Young Ahn and J. Dae Won, "A novel MRAS based speed sensorless control of induction motor," in *Industrial Electronics Society, 1999. IECON '99 Proceedings. The 25th Annual Conference of the IEEE*, 1999, pp. 933-938 vol.2.

[65] J. Holtz and J. Quan, "Sensorless vector control of induction motors at very low speed using a nonlinear inverter model and parameter identification," *Ieee Transactions on Industry Applications,* vol. 38, pp. 1087-1095, Jul-Aug 2002.

[66] L. Ben-Brahim, S. Tadakuma, and A. Akdag, "Speed control of induction motor without rotational transducers," *Ieee Transactions on Industry Applications,* vol. 35, pp. 844-850, Jul-Aug 1999.

[67] B. Karanayil, M. F. Rahman, and C. Grantham, "An implementation of a programmable cascaded low-pass filter for a rotor flux synthesizer for an induction motor drive," in *Power Electronics Specialists Conference, 2002. pesc 02. 2002 IEEE 33rd Annual*, 2002, pp. 1965-1970.

[68] G. Qiang, C. S. Spiteri, G. M. Asher, and M. Sumner, "Sensorless speed operation of cage induction motor using zero drift feedback integration with MRAS observer," in *Power Electronics and Applications, 2005 European Conference on*, 2005, pp. 9 pp.-P.9.

[69] C. Lascu, I. Boldea, and F. Blaabjerg, "A modified direct torque control for induction motor sensorless drive," *Ieee Transactions on Industry Applications,* vol. 36, pp. 122-130, Jan-Feb 2000.

[70] I. Kumara, "Speed sensorless field oriented control for induction motor drive," PhD, Newcastle University, Newcastle upon Tyne, UK, 2006.

[71] S. H. Kim, T. S. Park, J. Y. Yoo, G. T. Park, and N. J. Kim, "Dead time compensation in a vector-controlled induction machine," in *Power Electronics Specialists Conference, 1998. PESC 98 Record. 29th Annual IEEE*, 1998, pp. 1011-1016 vol.2.

[72] J. Campbell and M. Sumner, "Practical sensorless induction motor drive employing an artificial neural network for online parameter adaptation," *Iee Proceedings-Electric Power Applications,* vol. 149, pp. 255-260, Jul 2002.

[73] V. Vasic, S. N. Vukosavic, and E. Levi, "A stator resistance estimation scheme for speed sensorless rotor flux oriented induction motor drives," *Ieee Transactions on Energy Conversion,* vol. 18, pp. 476-483, Dec 2003.

[74] L. Zhen and L. Y. Xu, "Sensorless field orientation control of induction machines based on a mutual MRAS scheme," *Ieee Transactions on Industrial Electronics,* vol. 45, pp. 824-831, Oct 1998.

[75] V. Vasic and S. Vukosavic, "Robust MRAS-based algorithm for stator resistance and rotor speed identification," *IEEE Power Engineering Review,* vol. 21, pp. 39-41, 2001.

[76] H. Kubota and K. Matsuse, "Speed Sensorless Field-Oriented Control of Induction-Motor with Rotor Resistance Adaptation," *Ieee Transactions on Industry Applications,* vol. 30, pp. 1219-1224, Sep-Oct 1994.

[77] M. Shiref, "Investigation into PI controller output ripple in MRAS based electrical drives," Ph.D., Electrical and Electronic Engineering, Newcastle University
2013.

[78] A. Linder and R. Kennel, "Direct model predictive control-a new direct predictive control strategy for electrical drives," in *Power Electronics and Applications, 2005 European Conference on*, 2005, pp. 10 pp.-P. 10.

[79] S. Saeidi and R. Kennel, "A novel algorithm for model predictive control of AC electric drives," in *Electric Drives Production Conference (EDPC), 2012 2nd International*, 2012, pp. 1-7.

[80] S. Kouro, P. Cortes, R. Vargas, U. Ammann, and J. Rodriguez, "Model Predictive Control-A Simple and Powerful Method to Control Power Converters," *Ieee Transactions on Industrial Electronics,* vol. 56, pp. 1826-1838, Jun 2009.

[81] J. Rodriguez, J. Pontt, C. A. Silva, P. Correa, P. Lezana, P. Cortes*, et al.*, "Predictive current control of a voltage source inverter," *Ieee Transactions on Industrial Electronics,* vol. 54, pp. 495-503, Feb 2007.

[82] F. X. Wang, Z. B. Zhang, S. A. Davari, R. Fotouhi, D. A. Khaburi, J. Rodriguez*, et al.*, "An Encoderless Predictive Torque Control for an Induction Machine With a Revised Prediction Model and EFOSMO," *Ieee Transactions on Industrial Electronics,* vol. 61, pp. 6635-6644, Dec 2014.

[83] C. S. Lim, E. Levi, M. Jones, N. Abd Rahim, and W. P. Hew, "FCS-MPC-Based Current Control of a Five-Phase Induction Motor and its Comparison with PI-PWM Control," *Ieee Transactions on Industrial Electronics,* vol. 61, pp. 149-163, Jan 2014.

[84] R. Vargas, J. Rodriguez, C. A. Rojas, and M. Rivera, "Predictive Control of an Induction Machine Fed by a Matrix Converter With Increased Efficiency and Reduced Common-Mode Voltage," *Ieee Transactions on Energy Conversion,* vol. 29, pp. 473-485, Jun 2014.

[85] P. L. Jansen and R. D. Lorenz, "A Physically Insightful Approach to the Design and Accuracy Assessment of Flux Observers for Field-Oriented Induction Machine Drives," *Ieee Transactions on Industry Applications,* vol. 30, pp. 101-110, Jan-Feb 1994.

[86] J. Rodriguez, P. Cortes, R. Kennel, and M. P. Kazrnierkowski, "Model predictive control -- a simple and powerful method to control power converters," in *Power Electronics and Motion Control Conference, 2009. IPEMC '09. IEEE 6th International*, 2009, pp. 41-49.

[87] A. Linder and R. Kennel, "Model Predictive Control for Electrical Drives," in *2005 IEEE 36th Power Electronics Specialists Conference*, 2005, pp. 1793-1799.

[88] R. Kennel, A. Linder, and M. Linke, "Generalized predictive control (GPC)-ready for use in drive applications?," in *Power Electronics Specialists Conference, 2001. PESC. 2001 IEEE 32nd Annual*, 2001, pp. 1839-1844 vol. 4.

[89] S. Muller, U. Ammann, and S. Rees, "New time-discrete modulation scheme for matrix converters," *Ieee Transactions on Industrial Electronics,* vol. 52, pp. 1607-1615, Dec 2005.

[90] P. CortÉs, J. RodrÍguez, P. Antoniewicz, and M. Kazmierkowski, "Direct Power Control of an AFE Using Predictive Control," *IEEE Transactions on Power Electronics,* vol. 23, pp. 2516-2523, 2008.

[91] P. Cortes, J. Rodriguez, D. Quevedo, and C. Silva, "Predictive Current Control Strategy with Imposed Load Current Spectrum," in *Power Electronics and Motion Control Conference, 2006. EPE-PEMC 2006. 12th International*, 2006, pp. 252-257.

[92] D. E. Quevedo and G. C. Goodwin, "Multistep optimal analog-to-digital conversion," *Ieee Transactions on Circuits and Systems I-Regular Papers,* vol. 52, pp. 503-515, Mar 2005.

[93] R. Vargas, P. Cortes, U. Ammann, J. Rodriguez, and J. Pontt, "Predictive control of a three-phase neutral-point-clamped inverter," *Ieee Transactions on Industrial Electronics,* vol. 54, pp. 2697-2705, Oct 2007.

[94] S. A. Davari, D. A. Khaburi, and R. Kennel, "An Improved FCS-MPC Algorithm for an Induction Motor With an Imposed Optimized Weighting Factor," *Ieee Transactions on Power Electronics,* vol. 27, pp. 1540-1551, Mar 2012.

[95] W.Leonhard, *Control of Electrical Drives*, Third ed. New York: Springer, 2001.

[96] "IEEE Standard Test Procedure for Polyphase Induction Motors and Generators," *IEEE Std 112-2004 (Revision of IEEE Std 112-1996),* pp. 0_1-79, 2004.

[97] Y. B. Zbede, S. M. Gadoue, and D. J. Atkinson, "Model Predictive MRAS Estimator for Sensorless Induction Motor Drives," *Ieee Transactions on Industrial Electronics,* vol. 63, pp. 3511-3521, Jun 2016.

[98] Y. N. Lin and C. L. Chen, "Adaptive pseudoreduced-order flux observer for speed sensorless field-oriented control of IM," *Ieee Transactions on Industrial Electronics,* vol. 46, pp. 1042-1045, Oct 1999.