On Extended Kalman Filters with Augmented State Vectors for the Stator Flux Estimation in SPMSMs

T.J. Vyncke, R.K. Boel and J.A.A. Melkebeek
Department of Electrical Energy, Systems and Automation (EESA)
Ghent University (UGent), Sint-Pietersnieuwstraat 41, B-9000 Gent, Belgium
phone: +32 (0)9 264 3442, fax: +32 (0)9 264 3582, e-mail: Thomas.Vyncke@UGent.be

***Abstract*—The demand for highly dynamic electrical drives, characterized by high quality torque control, in a wide variety of applications has grown tremendously during the past decades. Direct torque control (DTC) for permanent magnet synchronous motors (PMSM) can provide this accurate and fast torque control. When applying DTC the change of the stator flux linkage vector is controlled, based on torque and flux errors. As such the estimation of the stator flux linkage is essential. In the literature several possible solutions for the estimation of the stator flux linkage are proposed. In order to overcome problems associated with the integration of the back-emf, the use of state observers has been advocated in the literature. Several types of state observers have been conceived and implemented for PMSMs, especially the Extended Kalman Filter (EKF) has received much attention. In most reported applications however the EKF is only used to estimate the speed and rotor position of the PMSM in order to realize field oriented current control in a rotor reference frame. Far fewer publications mention the use of an EKF to estimate the stator flux linkage vector in order to apply DTC. Still the performance of the EKF in the estimation of the stator flux linkage vector has not yet been thoroughly investigated. In this paper the performance of the EKF for stator flux linkage is studied and simulated. The possibilities to improve the estimation by augmenting the state vector and the consequences of these alterations are explored. Important practical aspects for FPGA implementation are discussed.**

I. Introduction

The use of highly dynamic electrical drives in a wide variety of applications has increased steadily in recent years. Within this market AC machines, and recently especially permanent magnet synchronous machines (PMSM's), have obtained dominance due to their characteristics of high efficiency, high power density and reliability. These highly dynamic electrical drives have to provide accurate and fast torque control together with the highest possible efficiency.

Rotor flux field oriented control has become an industry standard to control the torque and flux levels of AC machines. For induction motors (IM's) direct torque control (DTC) was proposed as an alternative control strategy in [1] and became very popular in the past two decades [2]. DTC for induction machines is inherently motion-state sensorless. In the past decade several authors [3]–[6] have proposed ways to adapt DTC to work with PMSM's.

To derive the principles of Direct Torque Control (DTC) the equation for electromagnetic torque T of a surface PMSM:

$$T = \frac{3N_p}{2L_s} \left|\underline{\Psi}_f\right| \left|\underline{\Psi}_s\right| \sin\delta \qquad (1)$$

is considered, where δ denotes the load angle between the stator flux linkage $\underline{\Psi}_s$ and permanent magnet flux linkage $\underline{\Psi}_f$ vectors in the stationary $\alpha\beta$ frame. The number of pole pairs is denoted by N_p and L_s is the stator inductance. From (1) can be seen that for constant stator flux linkage, the torque is changed by changing the load angle δ. The stator flux vector can be changed by applying from the inverter the voltage vector with the most appropriate radial and tangential components.

The switching decision is based on the estimated torque $T = \frac{3}{2}N_p(\Psi_{s,\alpha}I_\beta - \Psi_{s,\beta}I_\alpha)$, the stator flux linkage magnitude $|\underline{\Psi}_s|$ and stator flux linkage angle θ_{Ψ_s}; which are all determined by the estimation of the stator flux linkage components $\Psi_{s,\alpha}$ and $\Psi_{s,\beta}$. Thus the stator flux linkage estimation is crucial for a correct operation of the drive, as shown in figure 1.

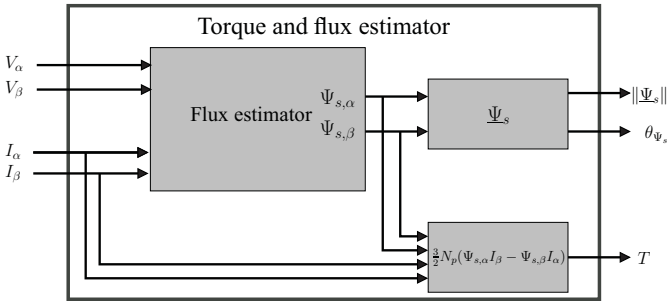

Figure 1: Torque and flux estimator schematic for DTC

Several estimation techniques have been reported in the literature [7]–[14]. Some include improvements on the back-emf integration such as low-pass-filtering [8] and stabilizing the integrator with a PI-corrector [12] or current offset [11]. Others use the current model of the PMSM, which often implies the use of a position sensor or needs an added, separate position estimation [14]; both are preferably avoided. State observers, such as the Extended Kalman Filter (EKF), that estimate the stator flux linkage vector by using its components as state variables or by calculating the flux components from other state components, and estimate rotor speed and position simultaneously, are another possibility. The EKF is often discussed for the sensorless control of PMSM's, however focused on the sensorless position estimation needed for field oriented control in the rotor flux reference frame. Few publications discuss the EKF for the estimation of the stator flux linkage vector [9], [10], [15]. Especially a thorough discussion addressing

the effect of incorrect parameters (resistance, inductance,rotor flux magnitude) lacks.

The paper is organized as follows: in section II the EKF is discussed, using two different sets of state variables for an SPMSM, and it is shown that estimation errors occur with incorrect motor parameters. In section III both EKF versions are expanded to include parameter estimations. Simulated results with these EKF implementations are given in section IV. Important aspects of the practical implementation for FPGA are discussed in section V.

II. Reduced-order EKF for Stator Flux Linkage Estimation

A. Stator Flux Linkage Estimation

In theory the integration of the back-emf can be used for this estimation when stator voltages and currents are measured:

$$\underline{\Psi}_s = \int_0^t (\underline{V}_s - R_s \underline{I}_s)dt + \underline{\Psi}_{s|t=0} \tag{2}$$

The use of a pure open-loop integration however has many disadvantages, as DC-offsets in the measurements make the integration drift and resistance variations decenter the estimation. Still it is a simple method, relying on only one parameter R_s and independent of the rotor position. An overview and comparison of several improved methods based on this principle is given in [15], where also current model based methods are discussed and compared.

The current model is defined in the stationary $\alpha\beta$ reference frame for an SPMSM by:

$$\Psi_{s,\alpha} = L_s I_\alpha + \Psi_f \cos(\theta) \tag{3}$$
$$\Psi_{s,\beta} = L_s I_\beta + \Psi_f \sin(\theta). \tag{4}$$

As is clear from equations (3-4), these methods are dependent on the rotor position θ, stator inductance L_s and permanent magnet flux Ψ_f. The resulting need for a position sensor is, especially in DTC which is an inherently position sensorless method, considered as a major disadvantage. Also the increased parameter dependence on the inductances is, considering the saturation, a disadvantage. To reduce the parameter dependence and to perform the rotor position estimation needed in the current model a state observer can be used. Several observers have been proposed in literature, a short overview is given in [7], [8]. In this paper the extended Kalman filter is selected for elaboration.

B. Reduced-order EKF

The Kalman filter is a stochastic recursive optimum-state estimator. For nonlinear systems an extended Kalman filter (EKF) can be used to obtain unmeasurable states (e.g. speed and rotor position) by using a model for the dynamical system, measured states and statistics of the system and measurement noise. By means of the noise input it is possible to take account of both measuring errors and modelling errors. The EKF is a two-step method as shown in figure 2. With the measured

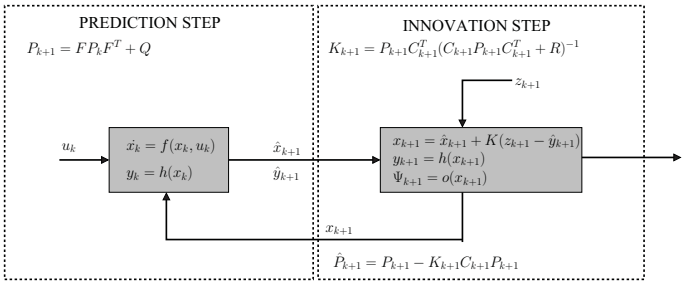

Figure 2: EKF scheme

inputs $\mathbf{u}_k$ and machine model ($\mathbf{f}(\mathbf{x}, \mathbf{u})$ and $\mathbf{h}(\mathbf{x})$) the next state of the machine $\hat{\mathbf{x}}_{k+1}$ is predicted (prediction step). From this state the next output $\hat{\mathbf{y}}_{k+1}$ is calculated and compared to the measured value $\mathbf{z}_{k+1}$. The error on the output, together with the covariance values of measurement noise $\mathbf{R}$ and system $\mathbf{Q}$ are used to correct the state values in the next step. Often the covariance matrices are chosen to be diagonal. In this correction or innovation step the Kalman gain matrix $\mathbf{K}_{k+1}$ is calculated as well.

In this paper two implementations of the EKF are studied. The same nonlinear state-space model for the PMSM is used, the difference between the two methods is based on the selection of the state variables. In EKFC the current components in the stationary reference frame are selected as state variables, as in [7], [9]. In EKFF the stator flux linkage components in the stationary reference frame are selected as state variables, as in [10]. This means that for EKFC the state vector consist of four measurable quantities, however due to the preference for a motion-state sensorless drive only two state variables are assumed to be measurable. For EKFF the state vectors then consists of four unmeasurable quantities.

In both cases the voltage $\mathbf{u} = [V_\alpha \ V_\beta]^T$ and current components in the stationary reference frame $\mathbf{y} = [I_\alpha \ I_\beta]^T$ are selected as input and output respectively. The EKF is of reduced order as the inertia is assumed to be infinite so that the mechanical equation is omitted. This is very advantageous as the load torque and inertia in the mechanical equation typically are not known. Because the speed ω is in the state vector the EKF will correct this modelling error if a good value is chosen for the covariance. More details about the tuning of EKF's can be found in [16].

In this paper we define, besides the output function $\mathbf{y}$ that is used to correct the estimation, an additional output function $\mathbf{o}(\mathbf{x})$ which expresses the 'useful' output (the stator flux components) as a function of the state components.

1) EKFC: EKF with current components:
The state vector $\mathbf{x}$ is chosen with the current components in the stationary reference frame as state variables as in [7], [9]

$$\mathbf{x} = [x_1 \ x_2 \ x_3 \ x_4]^T \tag{5}$$
$$= [I_\alpha \ I_\beta \ \omega \ \theta]^T, \tag{6}$$

where ω and θ denote rotor speed and position respectively. The system function $\mathbf{f}(\mathbf{x}, \mathbf{u})$, output function $\mathbf{h}(\mathbf{x})$ and Jacobians $\mathbf{F} = \frac{\mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}$ and $\mathbf{C} = \frac{\mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}$ are:

$$\mathbf{f}(\mathbf{x},\mathbf{u}) = \begin{bmatrix} -\frac{R_s}{L_s}x_1 + \frac{\Psi_f}{L_s}x_3\cos x_4 + \frac{u_1}{L_s} \\ -\frac{R_s}{L_s}x_2 + \frac{\Psi_f}{L_s}x_3\sin x_4 + \frac{u_2}{L_s} \\ 0 \\ x_3 \end{bmatrix} \quad (7)$$

$$\mathbf{h}(x) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \mathbf{o}(\mathbf{x}) = \begin{bmatrix} L_s x_1 + \Psi_f \cos x_4 \\ L_s x_2 + \Psi_f \sin x_4 \end{bmatrix} \quad (8)$$

$$\mathbf{F} = \frac{\mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}} = \begin{bmatrix} -\frac{R_s}{L_s} & 0 & \frac{\Psi_f}{L_s}\cos x_4 & -\frac{\Psi_f}{L_s}x_3\sin x_4 \\ 0 & -\frac{R_s}{L_s} & \frac{\Psi_f}{L_s}\sin x_4 & \frac{\Psi_f}{L_s}x_3\cos x_4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (9)$$

$$\mathbf{C} = \frac{\mathbf{h}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (10)$$

2) EKFF: EKF with flux components:

The state vector is chosen with the flux components in the stationary reference frame as state variables as in [10]

$$\mathbf{x} = [x_1\ x_2\ x_3\ x_4]^T \quad (11)$$

$$= [\Psi_{s,\alpha}\ \Psi_{s,\beta}\ \omega\ \theta]^T, \quad (12)$$

System and output functions are:

$$\mathbf{f}(\mathbf{x},\mathbf{u}) = \begin{bmatrix} -\frac{R_s}{L_s}x_1 + \frac{R_s}{L_s}\Psi_f\cos x_4 + u_1 \\ -\frac{R_s}{L_s}x_2 + \frac{R_s}{L_s}\Psi_f\sin x_4 + u_2 \\ 0 \\ x_3 \end{bmatrix} \quad (13)$$

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} \frac{x_1 - \Psi_f\cos x_4}{L_s} \\ \frac{x_2 - \Psi_f\sin x_4}{L_s} \end{bmatrix} \quad \mathbf{o}(\mathbf{x}) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (14)$$

$$\mathbf{F} = \frac{\mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}} = \begin{bmatrix} -\frac{R_s}{L_s} & 0 & 0 & -\frac{R_s\Psi_f}{L_s}\sin x_4 \\ 0 & -\frac{R_s}{L_s} & 0 & \frac{R_s\Psi_f}{L_s}\cos x_4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (15)$$

$$\mathbf{C} = \frac{\mathbf{h}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{1}{L_s} & 0 & 0 & \frac{\Psi_f}{L_s}\sin x_4 \\ 0 & \frac{1}{L_s} & 0 & \frac{\Psi_f}{L_s}\cos x_4 \end{bmatrix} \quad (16)$$

When inspecting the equations for EKFC and EKFF it is clear that the choice of state vector components for EKFF would appear as the more natural one (this is most obvious in $\mathbf{o}(\mathbf{x})$). Furthermore it is important to notice that the speed ω is not needed in the equations of EKFF (its use to estimate θ is not necessary), this means that we could further reduce the order of EKFF and omit $x_3 = \omega$ as a state variable and estimate θ directly. It is however retained for two reasons. Firstly because using the same order for EKFC and EKFF simplifies some practical implementation aspects as we can reuse matrix manipulations. Secondly because in an DTC drive the knowledge of the speed is advantageous for different control purposes (switching over from one voltage vector selection algorithm to another, select flux reference value). When inspecting the Jacobians $\mathbf{F} = \frac{\mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}$ and $\mathbf{C} = \frac{\mathbf{h}(\mathbf{x})}{\partial \mathbf{x}}$ it is obvious that the expression of $\mathbf{F}$ is more complicated for EKFC than EKFF while the reverse is true for $\mathbf{C}$.

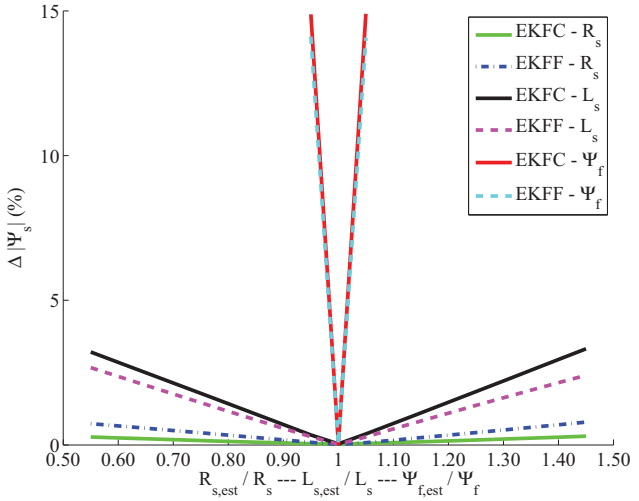

Figure 3: Sensitivity of EKFC and EKFF performance to parameter deviations

C. Influence of parameter variations

First the effect of an incorrect value for the stator resistance is evaluated. The simulations are done with the motor data found in appendix. The SPMSM is Direct Torque-Controlled and after a run-up it runs at half the rated speed. In figure 3 the RMS error is shown for the stator flux vector magnitude during steady state where $\frac{R_{s,est}}{R_s}$ is varied. Clearly the RMS error of the EKF estimators is very small, even for large deviations of R_s. In steady-state the RMS errors of the stator flux angle are also very small [15]. While only the steady state errors have been considered here, it has to be noted that wrong parameter values affect the (starting) transients even more, with errors that are even much larger than in steady state. The EKFs are methods based on the current model and thus also dependent on L_s and Ψ_f. In figure 3 the RMS-values of the errors are found. As shown in [15], the estimations with EKFC and EKFF do not yield better results than the open-loop current model with measured position. This means that the only remaining advantage is the sensorless fashion in which the estimation is executed. A thorough comparison between the performance of the EKFs (EKFC and EKFF) and several other stator flux estimators (most of them based on an integrator) is given in [15]. There the effects of parameter changes in R_s and L_s are discussed in more detail.

Clearly the EKF estimators can cope very well with errors in R_s, but variations in L_s and Ψ_f are more troublesome. The estimators remain stable but show a considerable steady-state deviation of both flux magnitude and angle, comparable to the case of on an open-loop current model (although one has to consider the fact that for an SPMSM the influence of L_s still is rather small). The correction in the estimation when R_s is varied is the result of the fact that the EKF estimators can correct for the modeling inaccuracies by the feedback loop. For variations in L_s and Ψ_f however this is not the case as the parameter L_s and Ψ_f, unlike R_s, are not only used

Name of EKF	Added state	**x**	**f(x,u)**	**h(x)**	**o(x)**
EKFCA1	L_s	$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} I_\alpha \\ I_\beta \\ \omega \\ \theta \\ \frac{1}{L_s} \end{bmatrix}$	$\begin{bmatrix} -R_s x_1 x_5 + \Psi_f x_3 x_5 \cos x_4 + x_5 u_1 \\ -R_s x_2 x_5 + \Psi_f x_3 x_5 \sin x_4 + x_5 u_2 \\ 0 \\ x_3 \\ 0 \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$	$\begin{bmatrix} \frac{x_1}{x_5} + \Psi_f \cos x_4 \\ \frac{x_2}{x_5} + \Psi_f \sin x_4 \end{bmatrix}$
EKFCA2	L_s, R_s	$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} I_\alpha \\ I_\beta \\ \omega \\ \theta \\ \frac{1}{L_s} \\ R_s \end{bmatrix}$	$\begin{bmatrix} -x_1 x_5 x_6 + \Psi_f x_3 x_5 \cos x_4 + x_5 u_1 \\ -x_2 x_5 x_6 + \Psi_f x_3 x_5 \sin x_4 + x_5 u_2 \\ 0 \\ x_3 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$	$\begin{bmatrix} \frac{x_1}{x_5} + \Psi_f \cos x_4 \\ \frac{x_2}{x_5} + \Psi_f \sin x_4 \end{bmatrix}$
EKFCA3	L_s, R_s, Ψ_f	$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} I_\alpha \\ I_\beta \\ \omega \\ \theta \\ \frac{1}{L_s} \\ R_s \\ \Psi_f \end{bmatrix}$	$\begin{bmatrix} -x_1 x_5 x_6 + x_3 x_5 x_7 \cos x_4 + x_5 u_1 \\ -x_2 x_5 x_6 + x_3 x_5 x_7 \sin x_4 + x_5 u_2 \\ 0 \\ x_3 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$	$\begin{bmatrix} \frac{x_1}{x_5} + x_7 \cos x_4 \\ \frac{x_2}{x_5} + x_7 \sin x_4 \end{bmatrix}$
EKFFA1	L_s	$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} \Psi_{s,\alpha} \\ \Psi_{s,\beta} \\ \omega \\ \theta \\ \frac{1}{L_s} \end{bmatrix}$	$\begin{bmatrix} -R_s x_1 x_5 + R_s \Psi_f x_5 \cos x_4 + u_1 \\ -R_s x_2 x_5 + R_s \Psi_f x_5 \sin x_4 + u_2 \\ 0 \\ x_3 \\ 0 \end{bmatrix}$	$\begin{bmatrix} x_5(x_1 - \Psi_f \cos x_4) \\ x_5(x_2 - \Psi_f \sin x_4) \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
EKFFA2	L_s, R_s	$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} \Psi_{s,\alpha} \\ \Psi_{s,\beta} \\ \omega \\ \theta \\ \frac{1}{L_s} \\ R_s \end{bmatrix}$	$\begin{bmatrix} -x_1 x_5 x_6 + \Psi_f x_5 x_6 \cos x_4 + u_1 \\ -x_2 x_5 x_6 + \Psi_f x_5 x_6 \sin x_4 + u_2 \\ 0 \\ x_3 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} x_5(x_1 - \Psi_f \cos x_4) \\ x_5(x_2 - \Psi_f \sin x_4) \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
EKFFA3	L_s, R_s, Ψ_f	$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} \Psi_{s,\alpha} \\ \Psi_{s,\beta} \\ \omega \\ \theta \\ \frac{1}{L_s} \\ R_s \\ \Psi_f \end{bmatrix}$	$\begin{bmatrix} -x_1 x_5 x_6 + x_5 x_6 x_7 \cos x_4 + u_1 \\ -x_2 x_5 x_6 + x_5 x_6 x_7 \sin x_4 + u_2 \\ 0 \\ x_3 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} x_5(x_1 - x_7 \cos x_4) \\ x_5(x_2 - x_7 \sin x_4) \end{bmatrix}$	$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Table I: Equations for EKFC and EKFF with augmented state vector

in $\mathbf{f}(\mathbf{x}, \mathbf{u})$. For EKFC the state vector $\mathbf{x}$ will converge to the correct values, but due to the use of L_s and Ψ_f in $\mathbf{o}(\mathbf{x})$ to determine Ψ_α and Ψ_β from $\mathbf{x}$ the output is incorrect. For EKFF L_s is used in $\mathbf{h}(\mathbf{x})$ and thus the state vector $\mathbf{x}$ will not converge to the right value.

III. Adding parameter estimations to the EKF

As demonstrated in the previous section, the EKFs fail to estimate the stator flux linkage vector correctly if certain motor parameters (those used in $\mathbf{h}(\mathbf{x})$ and $\mathbf{o}(\mathbf{x})$) deviate from the true values. One possibility to overcome this problem is to estimate the most important motor parameters in the EKF as well. This can be done by augmenting the state vector with the parameters to be estimated, where parameter variations are given no dynamics (i.e. the corresponding row of $\mathbf{f}(\mathbf{x},\mathbf{u})$ is 0).

In Table I the expressions are given for EKFC and EKFF with added parameter estimations. Three cases are considered. In the first case L_s is added to estimate as this is a parameter that, due to saturation, can vary strongly during operation of the drive. Furthermore it is present in either **h** or **o** and so errors in L_s propagate through the estimation. It has to be noted that in order to take the variation of L_s into account, actually $\frac{1}{L_s}$ is added to the state vector. This is advisable because in $\mathbf{f}(\mathbf{x}, \mathbf{u})$ (both for EKFC and EKFC) and $\mathbf{h}(\mathbf{x})$ (for EKFF) the stator inductance always is present as $\frac{1}{L_s}$. Choosing L_s as a state component would thus, due to the partial differentiation, result in mathematical expressions for the Jacobians **F** and **C** that are much more complex and could be prohibitive for an actual implementation. In the second case both R_s and L_s are estimated as R_s can vary greatly with temperature. Finally, in the third case, the estimation of all three relevant parameters L_s, R_s, Ψ_f is performed. The estimated values of the parameters can be used outside the EKF as well, e.g. in the control algorithm.

For the sake of brevity, the expressions for the Jacobians **F** and **C** have been omitted in Table I. However it is clear that the increased complexity of the augmented state vectors is even more easily seen in the first **F** and **C** compared to $\mathbf{f}(\mathbf{x}, \mathbf{u})$ and $\mathbf{h}(\mathbf{x})$, due to the increased non-linearity of the model. To this end it is useful to compare the resulting increase in complexity for the families of EKFCA and EKFFA filters. For EKFCA1 and EKFFA1 **F** and **C** are given in Table II.

Obviously every addition of a parameter to estimate (where we assume no dynamics) first of all results in a row of zeros for **F** both in EKFCA and EKFFA. More important however is the fact that the first two rows of **F** now contain additional elements which augment the computational load considerably. Furthermore some elements of **F** are now no longer constants, but are dependent on the added state variable.

The changes in **C** are very different for EKFCA1 and EKFFA1: for EKFCA1 only a column of zeros is added, where the expression for EKFFA1 now contains the added state variable and a column with elements which again increase the computational load.

This of course also means that the computation of the estimation covariance matrix **P** and the Kalman correction

EKFCA1

$$\mathbf{F} = \begin{bmatrix} -R_s x_5 & 0 & \Psi_f x_5 \cos x_4 & -\Psi_f x_5 x_3 \sin x_4 & -R_s x_1 + \Psi_f x_3 \cos x_4 + u_1 \\ 0 & -R_s x_5 & \Psi_f x_5 \sin x_4 & \Psi_f x_5 x_3 \cos x_4 & -R_s x_2 + \Psi_f x_3 \sin x_4 + u_2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

EKFFA1

$$\mathbf{F} = \begin{bmatrix} -R_s x_5 & 0 & 0 & -R_s \Psi_f x_5 \sin x_4 & R_s(-x_1 + \Psi_f \cos x_4) \\ 0 & -R_s x_5 & 0 & R_s \Psi_f x_5 \cos x_4 & R_s(-x_2 + \Psi_f \sin x_4) \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} x_5 & 0 & 0 & \Psi_f x_5 \sin x_4 & x1 - \Psi_f \cos x_4 \\ 0 & x_5 & 0 & \Psi_f x_5 \cos x_4 & x2 - \Psi_f \sin x_4 \end{bmatrix}$$

Table II: Expressions of **F** and **C** for EKFCA1 and EKFFA1 respectively

matrix **K** become increasingly complex (see figure 2) as the matrix operations (especially multiplication) become more computationally demanding with higher matrix sizes. For n_s state variables the size of the matrices is given in Table III. One important remark however is the fact that the matrix to invert (needed to calculate the Kalman gain) stays a 2×2 matrix independently from n_s. Obviously the computational effort to calculate the matrix $\mathbf{CPC}^T + \mathbf{R}$ however will strongly depend on n_s.

Matrix or array	Size		
$\mathbf{f}(\mathbf{x}, \mathbf{u})$	n_s	$\times$	1
$\mathbf{h}(\mathbf{x})$	2	$\times$	1
$\mathbf{Q}$	n_s	$\times$	n_s
$\mathbf{R}$	2	$\times$	2
$\mathbf{F}$	n_s	$\times$	n_s
$\mathbf{C}$	2	$\times$	n_s
$\mathbf{P}$	n_s	$\times$	n_s
$\mathbf{CPC}^T + \mathbf{R}$	2	$\times$	2
$\mathbf{K}$	n_s	$\times$	2

Table III: Matrix and array size for an EKF with n_s state vector components

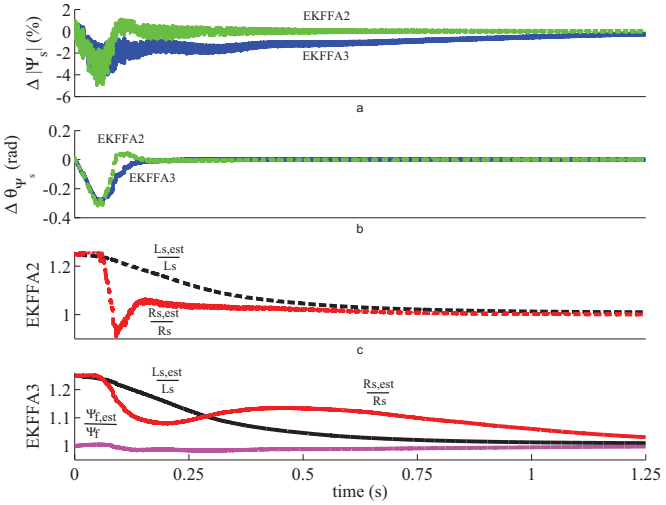

Figure 4: EKFFA2 and EKFFA3 with 25% error in R_s and L_s. a) Flux magnitude error b) flux angular error c) estimated parameters EKFFA2 d) estimated parameters EKFFA3

IV. Results in simulation

For the same motor and control scheme as before the behavior of the augmented EKFs is simulated. In figure 4a the flux amplitude error is shown for a low-dynamics drive cycle. The results shown are for EKFFA2 and EKFFA3, both with an initial 25% error on R_s and L_s and no error on Ψ_f. The angular error is shown in figure 4b and the evolution of the parameters in figure 4c and d. It is clear that EKFFA2 estimates both parameters correctly and results in good flux estimations. EKFFA3 should be able to cope with the errors in a similar way, but now also corrects Ψ_f during the transient and R_s only to a lesser extent. This leads to the observed drop in estimated flux magnitude in 4a. Similar observations are obtained with the other formulations of the filter : adding parameters to the estimation can work, but great care should be taken as unwanted cross coupling effects of parameter variations can occur with a poorly chosen covariance matrix **Q**. This further complicates the tuning of the EKF, which is often reported as one of the major drawbacks of this state observer (see also [16], where the problem for a 'standard' PMSM EKF is discussed). When implementing an EKF, one should carefully consider which parameters are the most likely to vary and at what rate. Clearly it is important to refrain from putting too little confidence (high values in **Q**) in the model

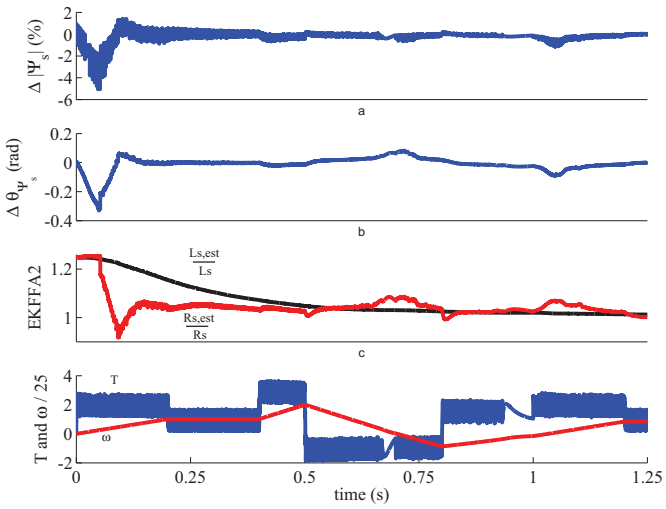

Figure 5: Performance of EKFFA2 with 25% error in R_s and L_s, dynamic drive cycle. a) Flux magnitude error b) flux angular error c) estimated parameters d) torque and speed (scaled by 25)

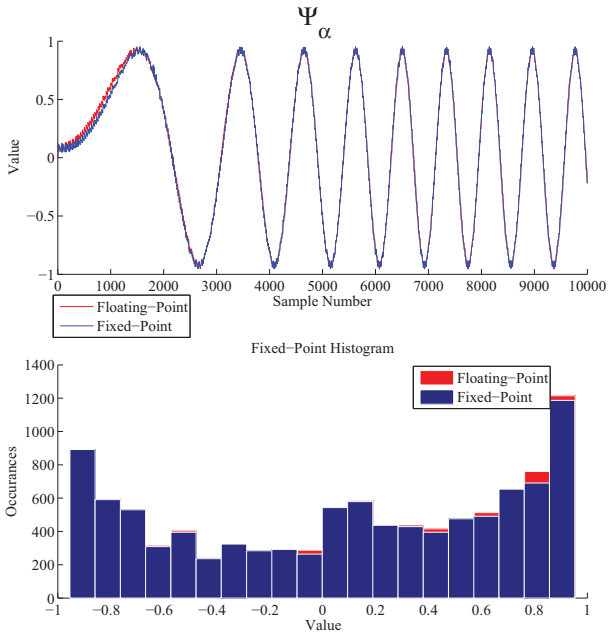

Figure 6: Estimation of the stator flux linkage component $\Psi_{s,\alpha}$ with the Matlab algorithm in floating-point and in fixed-point format by AccelDSP to implement it in the FPGA

as this could induce overcompensation of the parameters and thus result in poor performance.

In figure 5 the performance of EKFFA2 with the same initial errors as before is shown in a highly dynamic drive cycle. It is clear that with a good choice of covariance matrix and no additional erroneous parameters EKFFA2 offers good estimations.

V. Practical aspects for FPGA implementation

A. Per unit formulation and covariance matrices

Normalizing the state vector and the equations not only allows an easier conversion to fixed-point format, it also allow an easier setting of the covariance matrices as discussed in [16]. A per unit system with base quantities $V_b, I_b, \omega_b, \theta_b$ is used here. The base system can be selected in such a way that the state components always are smaller than 1 so that purely fractional fixed-point arithmetic can be used for most operations. However it does not ensure that the intermediate results (especially those resulting from the matrix inversion) stay smaller than 1. The flexibility offered by the FPGA and the Xilinx tools to program it however allow to cope with this.

All of the EKFs are initialized with a zero matrix for $\mathbf{P}$, while the covariance matrices are for EKFC and EKFF respectively:

$$\mathbf{Q} = diag(0.0012 \quad 0.0012 \quad 0.015 \quad 0.02)\ \mathbf{R} = diag(0.1 \quad 0.1) \tag{17}$$

$$\mathbf{Q} = diag(0.0027 \quad 0.0027 \quad 0.05 \quad 0.1)\ \mathbf{R} = diag(0.1 \quad 0.1) \tag{18}$$

These have been selected based on the method discussed in [16] and refined by the results from simulation. For the groups of EKFCA and EKFFA additional elements have to be selected to estimate the parameters. As said before, this can be tricky. Up till now they have been used as additional tuning parameters, but a method that expands the procedure of [16] would be desirable.

B. Implementation using AccelDSP

For the digital implementation of the stator flux linkage estimators two evaluation boards from Digilent Inc. are considered. As a rather low-cost option the Spartan 3E Starter Board, based on the Xilinx Spartan 3E FPGA (500K gates) clocked at 50MHz, is chosen. The Virtex II Pro board based on the XC2VP30 with a 32 or 100 MHz clock is considered here as a high-performance option. One goal is to optimize the implementation of EKFC and EKFF to the degree that it can easily be implemented on the Spartan 3E with a sampling frequency of 20kHz and enough resources left to implement the rest of the control (DTC in this case). Another goal is to implement the EKFs with augmented state vector. Due to the higher complexity in this case the specifications of a Spartan 3E board could be too limited (due to the increased degree of non-linearity it becomes increasingly harder to realize all the calculations with the 20 dedicated multipliers of the Spartan 3E). For the exploration of the possibilities to implement these EKFS the Virtex II Pro board is used.

For both options the configuration of the FPGA is programmed in Matlab/Simulink with the System Generator tool from Xilinx. Some specific functions are written in VHDL and interfaced through the Black Box block. The EKF algorithm is implemented with the AccelDSP tool from Xilinx. Here the implementation is briefly discussed. The AccelDSP tool takes a tested Matlab m-file with the algorithm to be implemented

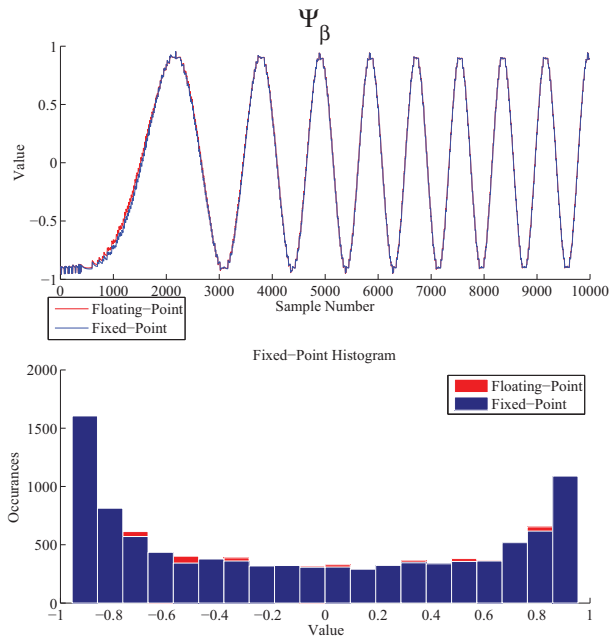

Figure 7: Estimation of the stator flux linkage component $\Psi_{s,\beta}$ with the Matlab algorithm in floating-point and in fixed-point format by AccelDSP to implement it in the FPGA

and assists the programmer during the conversion to a fixed-point version as a first step, the realisation of an RTL version as a second step and finally the creation of an HDL module or System Generator block.

In figures 6 and 7 the results are shown for the AccelDSP fixed-point implementation of EKFF. The loss of precision during the transition from floating-point to fixed-point format can be noticed but is rather small. Most importantly the round-off errors during the matrix operations do not result in instability of the EKF. More background on the detrimental effects of round-off errors on the performance of Kalman filters can be found in [17]. In this research it was attempted to retain enough precision to avoid these effects, whilst keeping the data types small enough to be implemented. Here the strength of the AccelDSP tool comes into play, as we can adjust the data type as desired for every mathematical operation.

For a 'naive' implementation, where little attention is given to the optimization of the implementation, Table IV gives the needed clock cycles, FPGA slices and embedded 18x18 multipliers when realized by AccelDSP for the Virtex II Pro. Given the number of slices needed, this version can not be implemented on the Spartan 3E board. This EKF can run in under 15 μs which is sufficiently fast for the proposed sampling frequency of 20kHz. However faster results can be obtained as in [18], also on a Virtex II Pro board.

When more attention is given to the optimization (re-using of calculated values) both a smaller and faster implementation is obtained, the results are given in Table V. Still the number of slices and especially the number of embedded multipliers is too high to be implemented on Spartan 3E (maximum values

Matrix or array	no. cycles	slices	18x18 mult.
Prediction of state vector	24	3297	5 (4%)
Prediction of covariance matrix	139	1201	12 (9%)
Calculation of the argument of the matrix invert	99	1392	12 (9%)
Matrix invert	85	2069	18 (13%)
Kalman gain calculation and update of covariance matrix	90	1372	14 (10%)
Innovation of state vector	84	1283	18 (13%)
clock cycles	521		
total slices		8545 (62%)	

Table IV: Number of FPGA clock cycles and slices needed per EKF module for EKFF on Virtex II Pro, 'naive' implementation clocked at 32 MHz

Matrix or array	clock cycles	slices	18x18 mult.
Prediction step	47	2978	23 (17%)
Kalman gain calculation	138	2596	24 (18%)
Innovation of state vector	10	1958	18 (13%)
clock cycles	195		
total slices		7532 (55%)	

Table V: Number of FPGA clock cycles and slices needed for EKFF on Virtex II Pro, better implementation

are 4658 and 20 for the Spartan 3E compared to 13696 and 136 for the Virtex II Pro). Clearly an improvement can be made by rolling some operations (in this implementation matrix multiplications are fully unrolled).

Besides the Spartan 3E implementation, the implementation of the augmented EKFs has to be explored. The FPGA implementation of the matrix invert can be re-used in all EKFCA and EKFFA versions, as the size of the matrix to invert is always 2×2. This means that the computational effort for this particular part of the algorithm will remain the same. However, the other computational effort needed for the other parts depends heavily on the value of n_s. Due to the increased non-linearity of the augmented versions the cycle time for calculations with **F** (and for the EKFF-family **C** as well) increases heavily. This is the result of the fact that not only the size increases but also more elements within the matrix **F** or **C** will be variable as constants are replaced by state components at several positions in the matrix. This implies that more dedicated multipliers will be used and less optimization by the AccelDSP tool (for example changing multiplications by constants to shift operations) will be performed.

Given the results in Table IV and V we can expect that the implementation of augmented extended kalman filters for Virtex II Pro should be possible, but the feasibility to do this on the Spartan 3E is not so certain.

C. Improvements for the FPGA implementation

In order to optimize the FPGA implementation (and to fit the implementation on a Spartan 3E) a further optimization of the fixed-point data format used in the calculations is required. Further improvements can be made by assuming that the Kalman gain **K** and the covariance matrix **P** are symmetrical, resulting in a significant reduction of the elements that need to be calculated. A further improvement is the use of RAM-blocks to store the different values during a calculation cycle

instead of calculating them at several positions in separate matrices. Depending on the resources needed for rest of the control algorithm and the desired cycle period the EKF calculations could be further rolled or unrolled in AccelDSP to optimize either the number of FPGA cells or the number of clock cycles needed. Especially for the implementation on a Spartan 3E the number of multipliers used should be kept under control.

VI. Conclusions

In this paper it is shown that the Extended Kalman Filter, although stable in the case of parameter variations, will produce a stator flux linkage vector estimation that deviates strongly if incorrect motor parameters are used. Three cases can be considered. The first case is the one where the wrong value of a parameter of only R_s is not a large problem as there are no additional errors in calculating the output, so that the feedback correction loop is able to handle this. In the second case, which occurs in EKFF, a wrong value for L_s or Ψ_f would, even for a correct state estimate, lead to an incorrect output $\mathbf{y}$ and thus correction. The third case occurs in the EKFC: state vector $\mathbf{x}$ and output $\mathbf{y}$ can be correct, but an incorrect flux estimate is obtained through $\mathbf{o}(\mathbf{x})$.

To mitigate this problem several formulations of the EKF are given where the state vector is augmented with the parameters that need to be estimated. When implementing these, great care should be taken. It is shown that an uncareful selection of the parameters to be estimated and their covariance elements results in strongly divergent EKFs. If a good choice is made for the covariance, or if additional information about some of the parameters is available, a high quality estimation can be obtained.

Some aspects and caveats of the FPGA implementation are discussed. Specifically the process of translating the floating-point Matlab algorithm to an HDL or System Generator module by using the AccelDSP tool from Xilinx is addressed.

Acknowledgment

T. Vyncke wishes to thank the Research Foundation-Flanders for his grant as Ph. D. fellowship of the Research Foundation - Flanders (FWO).

This work was supported by the GOA project BOF 07/GOA/006. The research was performed as part of the Interuniversity Attraction Poles programme IUAP P6/21 financed by the Belgian government.

Appendix

R_s	2.875 Ω	L_s	8.5 mH	J	0.008 kgm^2
Ψ_f	0.175 Wb	N_p	4	F	0.001 Nms

Parameters of SPMSM used in simulation

References

[1] I. Takahashi and T. Noguchi, "A new quick-response and high-efficiency control strategy of an induction motor," *IEEE Trans. Ind. Applicat.*, vol. 22, no. 5, pp. 820–827, Sept./Oct. 1986.

[2] G. S. Buja and M. P. Kazmierkowski, "Direct torque control of PWM inverter-fed AC motors – a survey," *IEEE Trans. Ind. Electron.*, vol. 51, no. 4, pp. 744–757, Aug. 2004.

[3] L. Zhong, M. F. Rahman, W. Y. Hu, and K. W. Lim, "Analysis of direct torque control in permanent magnet synchronous motor drives," *IEEE Trans. Power Electron.*, vol. 12, no. 3, pp. 528–536, May 1997.

[4] M. Niemelä, J. Luukko, and J. Pyrhönen, "Position sensorless PMSM DTC-drive for industrial applications," in *Conf. Proc. EPE*, Graz, 2001, p. 10.

[5] M. F. Rahman, L. Zhong, M. E. Haque, and M. Rahman, "A direct torque-controlled interior permanent-magnet synchronous motor drive without a speed sensor," *IEEE Trans. Energy Conversion*, vol. 18, no. 1, pp. 17–22, Mar. 2003.

[6] D. Swierczynski and M. P. Kazmierkowski, "Direct torque control of permanent magnet synchronous motor (PMSM) using space vector modulation (DTC-SVM) – simulation and experimental results," in *Conf. Proc. IEEE 28th Annual Conference of the Industrial Electronics Society (IECON'02)*, vol. 1, Nov. 5–8, 2002, pp. 751–755.

[7] P. Vas, *Sensorless Vector and Direct Torque Control*. New York: Oxford University Press, 1998, pp. 122–178.

[8] M. F. Rahman, M. E. Haque, L. Tang, and L. Zhong, "Problems associated with the direct torque control of an interior permanent-magnet synchronous motor drive and their remedies," *IEEE Trans. Ind. Electron.*, vol. 51, no. 4, pp. 799–809, Aug. 2004.

[9] A. Llor, J. Rétif, X. Lin-Shi, and S. Arnalte, "Direct stator flux linkage control technique for a permanent magnet synchronous machine," in *Conf. Rec. IEEE 34th Annual Power Electronics Specialists Conference (PESC'03)*, vol. 1, June 15–19, 2003, pp. 246–250.

[10] V. Comnac, M. Cernat, F. Moldoveanu, and I. Draghici, "Sensorless speed and direct torque control of surface permanent magnet synchronous machines using an extended kalman filter," in *Conf. Proc. 9th Mediterranean Conference on Control and Automation (MED'01)*, Dubrovnik, Croatia, June 27–29, 2001, p. 6.

[11] J. Luukko, M. Niemelä, and J. Pyrhönen, "Estimation of the flux linkage in a direct-torque-controlled drive," *IEEE Trans. Ind. Electron.*, vol. 50, no. 2, pp. 283–287, Apr. 2003.

[12] G. D. Andreescu and A. Popa, "Flux estimator based on integrator with DC-offset correction loop for sensorless direct torque and flux control," in *Proc. 15th Int. Conf. on Electrical Machines ICEM 2002*, Bruges, Belgium, Aug. 2002, p. 6.

[13] B. Lang, W. Liu, and G. Luo, "A new observer of stator flux linkage for permanent magnet synchronous motor based on kalman filter," in *Proc. 2nd IEEE Conf. on Industrial Electronics and Applications*, Harbin, China, May 2007, pp. 1813 – 1817.

[14] G. D. Andreescu, C. I. Pitic, F. Blaabjerg, and I. Boldea, "Combined flux observer with signal injection enhancement for wide speed range sensorless direct torque control of ipmsm drives," *IEEE Trans. Energy Conversion*, vol. 23, Issue 2, pp. 393 – 402, June 2008.

[15] T. J. Vyncke, R. K. Boel, and J. A. Melkebeek, "A comparison of stator flux linkage estimators for a direct torque controlled pmsm drive," in *IEEE Industrial Electronics Conference 2009*, Porto, Portugal, Nov.3-5 2009, pp. 967–974.

[16] S. Bolognani, L. Tubiana, and M. Zigliotto, "Extended Kalman filter tuning in sensorless PMSM drives," *IEEE Trans. Ind. Applicat.*, vol. 39, no. 6, pp. 1741–1747, Nov./Dec. 2003.

[17] M. S. Grewal and A. P. Andrews, *Kalman Filtering, Theory and Practice Using MATLAB*. New York: Wiley-Interscience, 2001, pp. 202–271.

[18] L. Idkhajine, E. Monmasson, and A. Maalouf, "Fully FPGA-based sensorless control for AC drive using an extended kalman filter," in *IEEE Industrial Electronics Conference 2009*, Porto, Portugal, Nov.3-5 2009, pp. 2939–2944.