Two-Phase Permanent Magnet Synchronous Motor State Estimation

Noaman M. Noaman

Nahrain University
College of engineering
Computer Engineering
Department
noaman961@gmail.com

Amjad J. Humaidi

University of Technology
Control and Systems
Department
Amjad_jalil2000@yahoo.com

Aid Qassim Hussein

College of Electrical and
Electronic Techniques
aid_qassim975@yahoo.com

Abstract

The goal of this paper is to estimate the states of two-phase permanent magnet (PM) synchronous motor. The system is highly nonlinear and one therefore cannot directly use any linear systems tools for estimation. However, if one can linearize the system around a nominal (possibly time-varying) operating point then linear system tools could be used for control and estimation. The standard discrete Kalman filter (KF) has been used for state estimation. As such, the nonlinear model has been discretized and extended to be suitably applied for such filter. The entire state estimated system has been modeled using MATLAB/SIMULINK blocks. The state estimation algorithm and the motor discretized model are coded inside special S-functions of m-file type. Also, the error covariance matrices of measurement and process will be developed from the system model.

Keywords:

State estimator, two-phase permanent magnet synchronous motor, linearized model, Extended Kalman Filter.

Introduction

In controlling AC machine drives, speed transducers such as tacho-generators, resolvers, or digital encoders are used to obtain speed information. Using these speed sensors has some disadvantages [1].

□ They are usually expensive,

□ The speed sensor and the corresponding wires will take up space,

□ In defective and aggressive environments, the speed sensor might be the weakest part of the system.

Especially the last item degrades the systems reliability and reduces the advantage of an induction motor drive system. This has led to a great many speed sensor less vector control methods.

On the other hand, avoiding sensor means use of additional algorithms and added computational complexity that requires high-speed processors for real time applications. As digital signal processors have become cheaper, and their performance greater, it has become possible to use them for controlling electrical drives as a cost effective solution [2].

Estimation of unmeasurable state variables is commonly called observation. A device (or a computer program) that estimates or observes the states is called a state-observer. An observer can be classified according to the type of representation used for the plant to be observed [1].

If the plant is deterministic, then the observer is a deterministic observer; otherwise it is a stochastic observer. The most commonly used observers are Luenberger and

Kalman types [2]. The Luenberger observer (LO) is of the deterministic type, and the Kalman Filter (KF) is of the stochastic type. The basic Kalman filter is only applicable to linear stochastic systems, and for non-linear systems the extended Kalman filter (EKF) can be used, which can provide estimates of the states of a system or of both the states and parameters [1,2].

The EKF is a recursive filter (based on the knowledge of statistics of both the state and noise created by measurement and system modeling), which can be applied to non-linear time varying stochastic systems. EKF being insensitive to parameter changes and used for stochastic systems where measurement and modeling noises are taken into account.

Model of Two Phase PM Synchronous Motor

The continuous-time electromechanical model of two-phase permanent magnet (PM) synchronous motor is fourth order, nonlinear and can be described by:

$$\begin{aligned}
\dot{i}_a &= -(R_a/L)\ i_a + (\lambda/L)\ \omega_r \sin\theta_r + (1/L)\ u_a \\
\dot{i}_b &= -(R_a/L)\ i_b - (\lambda/L)\ \omega_r \cos\theta_r + (1/L)\ u_b \\
\dot{\omega}_r &= -(3\lambda/2J)\ i_a \sin\theta_r + (3\lambda/2J)\ i_b \cos\theta_r \\
&\qquad - (F/L)\ \omega_r - (1/J)\ T_L \\
\dot{\theta}_r &= \omega_r
\end{aligned} \tag{1}$$

where i_a**,** and i_b are the currents through the two windings, R_a and L are the resistance and inductance of the windings, θ and ω_r are the angular position and velocity of the rotor, λ is the flux constant of the motor, u_a and u_b are the voltages applied across the two windings, J is the moment of inertia of the rotor and its load, F is the viscous friction of the rotor, and T_L is the load torque [3].

However, the system is highly nonlinear and one therefore cannot directly use any linear systems tools for estimation. However, if one can linearize the system around a nominal (possibly time-varying) operating point then linear system tools could be used for control and estimation. We start by defining a state vector as $\boldsymbol{x}=[i_a \quad i_b \quad \omega_r \quad \theta_r]^T$ and the output vector as $\mathbf{y}=[i_a \;\; i_b]^T$. With this definition, Eq.(1) can be written compactly as

$$\dot{\boldsymbol{x}} = [\dot{x}_1 \quad \dot{x}_2 \quad \dot{x}_3 \quad \dot{x}_4]^T$$

$$\dot{\boldsymbol{x}} = \boldsymbol{Ax} + \boldsymbol{Bu} + \boldsymbol{M} \; T_L \tag{2}$$

$$\mathbf{y= C \; x} \tag{3}$$

where

$$\boldsymbol{A} = \begin{bmatrix} -\frac{R}{L} & 0 & \frac{\lambda}{L}\sin x_4 & 0 \\ 0 & -\frac{R}{L} & -\frac{\lambda}{L}\cos x_4 & 0 \\ -\frac{3\lambda}{2J}\sin x_4 & \frac{3\lambda}{2J}\cos x_4 & -\frac{F}{J} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{B}=\begin{bmatrix} (1/L) & 0 \\ 0 & (1/L) \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{M}=\begin{bmatrix} 0 \\ 0 \\ 0 \\ -\frac{1}{J} \end{bmatrix} \text{ and } \mathbf{C}=\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^{\boldsymbol{T}}$$

The motor Equation (2) is to be discretized for the digital implementation as:

$$x_{k+1} = A_k x_k + B_k u_k + M_k \; T_L \tag{4}$$

$$\boldsymbol{y_k} = \boldsymbol{C_k x_k} \tag{5}$$

$\boldsymbol{A_k}$ and $\boldsymbol{B_k}$ are the discretizd systems and input matrices, respectively. They are [1, 3, 4]:

$$\boldsymbol{A_k} = \boldsymbol{e}^{\boldsymbol{AT}} = \boldsymbol{I} + \boldsymbol{AT} + \frac{(\boldsymbol{AT})^2}{2!} + ...$$

$$\cong \boldsymbol{I} + \boldsymbol{AT} \tag{6}$$

$$\boldsymbol{B_k} = \int_0^{\boldsymbol{T}} \boldsymbol{e}^{\boldsymbol{A}\zeta} \boldsymbol{B} d\zeta = \left[\boldsymbol{e}^{\boldsymbol{AT}} - \boldsymbol{I}\right] \boldsymbol{A}^{-1}\boldsymbol{B}$$

$$= \boldsymbol{BT} + \frac{\boldsymbol{ABT}^2}{2!} + \ldots \cong \boldsymbol{BT} \tag{7}$$

$$\boldsymbol{M_k} \cong \boldsymbol{M}\mathrm{T} \tag{8}$$

$$\boldsymbol{C_k} = \boldsymbol{C} \tag{9}$$

where T is the sampling time and **I** is an identity (4×4) matrix. The above approximation is justified due to the small size of sampling time and the presence of increasingly large factorials, which further diminishes the magnitude of the higher-order terms.

$$A_k = \begin{bmatrix} a_{11} & 0 & a_{13}\sin x_4(k) & 0 \\ 0 & a_{11} & -a_{13}\cos x_4(k) & 0 \\ a_{31}\sin x_4(k) & -a_{31}\cos x_4(k) & a_{33} & 0 \\ 0 & 0 & a_{43} & 1 \end{bmatrix}$$

$$\boldsymbol{B_k} = \begin{bmatrix} b_{11} & 0 \\ 0 & b_{11} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \text{ and } \boldsymbol{M_d} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -\dfrac{T}{J} \end{bmatrix}$$

where

$$a_{11} = 1 - \frac{RT}{L},\ a_{13} = \frac{\lambda T}{L},\ a_{31} = -\frac{3\lambda T}{2J}$$

$$a_{33} = 1 - \frac{FT}{J},\ a_{43} = T \text{ and } b_{11} = \frac{T}{L}$$

If the noises Δu_a , Δu_b have corrupted the inputs u_a and u_b, respectively, and the noise $\Delta\alpha$ has been admitted to account for uncertainties in the load torque, then a noise vector will arise in Eq.(4)

$$\boldsymbol{x_{k+1}} = \boldsymbol{A_k}\boldsymbol{x_k} + \boldsymbol{B_k}\begin{bmatrix} u_a + \Delta u_a \\ u_b + \Delta u_b \end{bmatrix}_k + \boldsymbol{M_d}\ (T_L + \Delta T_L)$$

$$\boldsymbol{x_{k+1}} = \boldsymbol{A_d}\boldsymbol{x_k} + \boldsymbol{B_k}\boldsymbol{u_k} + \boldsymbol{M_d}\ T_L + \boldsymbol{w_k} \tag{10}$$

where

$$\boldsymbol{w_k} = \begin{bmatrix} (T/L)\Delta u_a \\ (T/L)\Delta u_b \\ -(T/J)\Delta T_L \\ 0 \end{bmatrix} \tag{11}$$

Similarly, if the measurements i_a and i_b are distorted by noises Δi_a and Δi_b respectively, then Eq.(5) becomes

$$\boldsymbol{y_k} = \boldsymbol{C_d}\begin{bmatrix} i_a + \Delta i_a \\ i_b + \Delta i_b \end{bmatrix} = \boldsymbol{C_d}\,\boldsymbol{x_k} + \boldsymbol{C_d}\begin{bmatrix} \Delta i_a \\ \Delta i_b \end{bmatrix}$$

$$= \boldsymbol{C_d}\,\boldsymbol{x_k} + \begin{bmatrix} \Delta i_a \\ \Delta i_b \end{bmatrix} = \boldsymbol{C_d}\,\boldsymbol{x_k} + \boldsymbol{v_k} \tag{12}$$

where

$$\boldsymbol{v_k} = \begin{bmatrix} \Delta i_a \\ \Delta i_b \end{bmatrix} \tag{13}$$

where the vectors $\boldsymbol{w_k}$ and $\boldsymbol{v_k}$ are called the process and measurement noises respectively.

The Kalman filter theory and algorithm

The aim in all estimation problems is to have an estimator that gives an accurate estimate of the true state even though one cannot directly measure it. Two obvious requirements should be attained [5]:

❑ First, the average value of our state estimate is to be equal to the average value of the true state. That is, the estimate has not to be biased one way or another.

Mathematically, one would say that the expected value of the estimate should be equal to the expected value of the state.

- Second, the requirement a state estimate that varies from the true state as little as possible. That is, not only do we want the average of the state estimate to be equal to the average of the true state, but also want an estimator that results in the smallest possible variation of the state estimate. Mathematically, an estimator with the smallest possible error variance is sought.

It so happens that the Kalman filter is the estimator that satisfies these two criteria. But the Kalman filter solution does not apply unless certain assumptions about the noise that affects the system under study must be satisfied:

1. It is firstly to assume that the average value of both $\boldsymbol{w_k}$ and $\boldsymbol{v_k}$ are zero.
2. One has to further assume that no correlation exists between $\boldsymbol{w_k}$ and $\boldsymbol{v_k}$. That is, at any time *k*, $\boldsymbol{w_k}$ and $\boldsymbol{v_k}$ are independent random variables. Then the noise covariance matrices $\boldsymbol{S_w}$ and $\boldsymbol{S_v}$ are defined as:

Process noise covariance:

$$\boldsymbol{S_w} = \boldsymbol{E}(\boldsymbol{w_k}\boldsymbol{w_k^T}) \tag{13}$$

Measurement noise covariance:

$$\boldsymbol{S_v} = \boldsymbol{E}(\boldsymbol{v_k}\boldsymbol{v_k^T}) \tag{14}$$

where $\boldsymbol{w^T}$ and $\boldsymbol{v^T}$ indicate the transpose of $\boldsymbol{w}$ and $\boldsymbol{v}$ random noise vectors, and E(.) means the expected value.

Substituting Eq.(10) into Eq.(13), and Eq.(12) into Eq.(14), one can get the following process and measurement noise covariance matrices:

$$\boldsymbol{s}_{\boldsymbol{w}} = \quad E\left(\begin{bmatrix} \left(\frac{T}{L}\right)^2 (\Delta u_a)^2 & \left(\frac{T}{L}\right)^2 \Delta u_a \Delta u_b & \left(\frac{T^2}{JL}\right)\Delta u_a \Delta T_L & 0 \\ \left(\frac{T}{L}\right)^2 \Delta u_a \Delta u_b & \left(\frac{T}{L}\right)^2 (\Delta u_b)^2 & \left(\frac{T^2}{JL}\right)\Delta u_b \Delta T_L & 0 \\ \left(\frac{T^2}{JL}\right)\Delta u_a \Delta T_L & \left(\frac{T^2}{JL}\right)\Delta u_b \Delta T_L & \left(\frac{T}{J}\right)^2 (\Delta T_L)^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\right) \tag{15}$$

$$\boldsymbol{s}_{\boldsymbol{v}} = \quad E\left(\begin{bmatrix} (\Delta i_a)^2 & \Delta i_a \Delta i_b \\ \Delta i_a \Delta i_b & (\Delta i_b)^2 \end{bmatrix}\right) \tag{16}$$

If the noises $\Delta u_a\,(\Delta u_b), \Delta T_L, \Delta i_a\,(\Delta i_b)$ are white, zero mean, uncorrelated, and have known variances σ_i^2, σ_T^2 and σ_M^2, respectively, then the covariance matrices $\boldsymbol{s}_{\boldsymbol{w}}$ and $\boldsymbol{s}_{\boldsymbol{v}}$ will become

$$\boldsymbol{s}_{\boldsymbol{w}} = \begin{bmatrix} \left(\frac{T}{L}\right)^2 \sigma_i^2 & 0 & 0 & 0 \\ 0 & \left(\frac{T}{L}\right)^2 \sigma_i^2 & 0 & 0 \\ 0 & 0 & \left(\frac{T}{J}\right)^2 \sigma_T^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{17}$$

$$\boldsymbol{s}_{\boldsymbol{v}} = \begin{bmatrix} \sigma_M^2 & 0 \\ 0 & \sigma_M^2 \end{bmatrix} \tag{18}$$

One may summarize the recursive state estimation of the discrete Kalman filter as shown in Fig.(1). In the figure, the superscripts "-1", "T", "+" and "-" indicate matrix inversion, matrix transposition, posteriori and priori of variable respectively. The **K** matrix is called the Kalman gain and the **P** matrix is called the estimation error covariance. The flowchart includes the initialization of state $\hat{x}_0$ in the absence of any observed data at *k=0,* and the initial value of the a posteriori covariance matrix $\boldsymbol{P}_0$ [6].

The timing diagram of the various quantities involved in the discrete optimal filter equations is shown in Fig.(2). The figure shows that after we process the measurement at time (*k-1*), we have an estimate of $\boldsymbol{x}_{\boldsymbol{k}+1}$ (denoted $\hat{\boldsymbol{x}}_{\boldsymbol{k}-1}^{+}$) and the covariance of that estimate (denoted $\boldsymbol{P}_{\boldsymbol{k}-1}^{+}$). When time *k* arrives, before we process the measurement at time *k* we compute an estimate of $\boldsymbol{x}_{\boldsymbol{k}}$ (denoted $\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{-}$*)* and the covariance of that estimate (denoted $\boldsymbol{P}_{\boldsymbol{k}}^{-}$*)*. Then the measurement is processed at time *k* to refine our estimate of $\boldsymbol{x}_{\boldsymbol{k}}$. The resulting estimate of $\boldsymbol{x}_{\boldsymbol{k}}$ is denoted $\hat{\boldsymbol{x}}_{\boldsymbol{k}}^{+}$ and its covariance is denoted $\boldsymbol{P}_{\boldsymbol{k}}^{+}$.

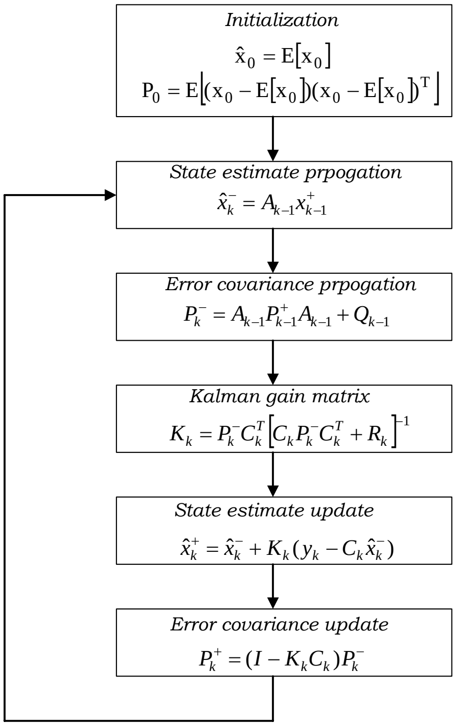
Initialization
$\hat{x}_0 = E[x_0]$
$P_0 = E[(x_0 - E[x_0])(x_0 - E[x_0])^T]$
State estimate prpogation
$\hat{x}_k^- = A_{k-1}x_{k-1}^+$
Error covariance prpogation
$P_k^- = A_{k-1}P_{k-1}^+A_{k-1} + Q_{k-1}$
Kalman gain matrix
$K_k = P_k^- C_k^T \left[C_k P_k^- C_k^T + R_k\right]^{-1}$
State estimate update
$\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k - C_k\hat{x}_k^-)$
Error covariance update
$P_k^+ = (I - K_kC_k)P_k^-$

Figure (1) Recursive Algorithm of Discrete Kalman Filter

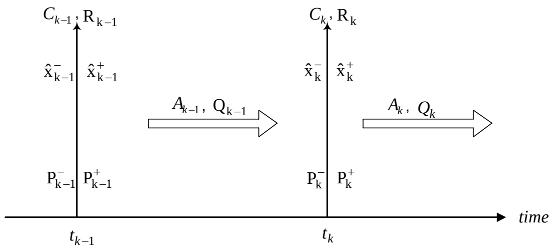

Figure (2) Timeline showing a priori and a posteriori state estimates and estimation-error covariance.

By substituting error covariance update equation into propagation equation, and the state estimate propagation equation into update equation, the algorithm of Fig.(1) will be summarized as [3,5,6-10]

$$K_k = P_k C_k^T \left[C_k P_k C_k^T + R_k\right]^{-1} \qquad (19)$$

$$\hat{x}_k = A_{k-1}\hat{x}_{k-1} + K_k(y_k - C_k\hat{x}_k) \qquad (20)$$

$$P_k = A_{k-1}((I - K_k C_k)P_k)A_{k-1} + Q_{k-1} \qquad (21)$$

Extended Kalman Filter (EKF)

The state-space model of Eqs.(10) and (12) can be rewritten in the following form:

$$\boldsymbol{x_{k+1}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{k}) + \boldsymbol{w_k} \qquad (22)$$

$$\boldsymbol{y_k} = \boldsymbol{C_d}\,\boldsymbol{x_k} + \boldsymbol{v_k} \qquad (23)$$

where

$$f(x,u,k) = \begin{bmatrix} a_{11}x_1(k) + a_{13}x_3(k)\sin x_4(k) + b_{11}u_a \\ a_{11}x_2(k) - a_{13}x_3(k)\cos x_4(k) + b_{11}u_b \\ a_{31}x_1(k)\sin x_4(k) - a_{31}x_2(k)\cos x_4(k) + a_{33}x_3(k) \\ a_{43}x_3(k) + x_4(k) \end{bmatrix}$$

It is clear that *f(x,u,k)* is nonlinear. However, to use nonlinear model with the standard KF, the model must be linearized about the current operating point, giving a linear perturbation model represented by Jacobian matrix $F(x,u,k)$,

$$F(x,u,k) = \frac{\partial f\,(x,u,k)}{\partial x}\bigg|_{\hat{x}(k)+u(k)} \tag{24}$$

$$= \begin{bmatrix} a_{11} & 0 & a_{13}\sin x_4(k) & a_{13}\,x_3(k)\cos x_4(k) \\ 0 & a_{11} & -a_{13}\cos x_4(k) & a_{13}\,x_3(k)\sin x_4(k) \\ a_{31}\sin x_4(k) & -a_{31}\cos x_4(k) & a_{33} & a_{31}(x_1(k)\cos x_4(k)+x_2(k)\sin x_4(k)) \\ 0 & 0 & a_{43} & 1 \end{bmatrix}$$

By now, the Jacobian matrix is replaced by A_k into Eq.s (20) and (21).

Modeling of Motor State Estimation System Using MATLAB/SIMULINK

SIMULINK is an extension to MATLAB and allows graphical block diagram modeling and simulation of dynamic systems. It is easier to develop state estimator using this package, as many components of the system are already included in the SIMULINK block diagram library.

The discretized model of the motor and the state estimation algorithm has been entered into a S-function-type of *m*-file. An *m*-file is a MATLAB program that allows algorithms or equations to be entered in a programming language. An S-function block, from the SIMULINK nonlinear library, links this m-file into a graphical block for use within the overall state estimation system.

Two quadrature sinusoidal waveforms drawn from the SIMULINK library have fed both the blocks of motor dynamic system and the state estimator, as shown in Fig.(3). The load change has been permitted and the repeating sequence block, from the

SIMULINK source library, is employed. The S-function block of motor model generates the actual states. The state estimator block receives, in addition to inputs u_a, u_b and T_L, the actual currents i_a and i_b. The estimator produces the estimated states of the motor.

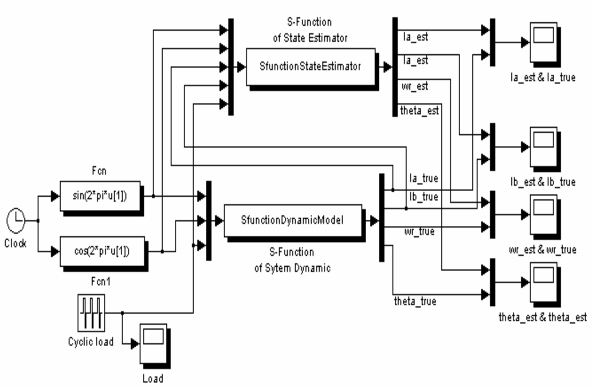

Figure (3) SIMULINK Modeling of Motor State Estimation System

Simulated Results

The parameters of the motor are listed in Appendix (I). The SIMULINK model of Fig.(3) has been run and the estimated and actual states representing stator currents and rotor velocity and position has shown in Fig.(4). The system was simulated at sampling time (T=2.5 ms). One can easily notice that the EKF estimator could successively estimate the motor states and the estimator showed an excellent noise rejection capability. However, one can observe that the estimator does hardly estimate the speed and the angular position at the motor starting, but there is a perfect overlap at steady state.

Figure (5) shows the outputs of the estimator when the sampling time is increased to (T=2.95 ms). The performance of estimator will show a great degradation in its responses. This unstable behavior of the estimator is attributed to matrix singularity problems in the Kalman gain matrix. As this would assign the gain K large values, which will reflect directly to updated state estimates.

In Figure (6), a mechanical load having the waveform of Fig.(7) has been applied. It is clear from the figure that the estimated speed state still well tracks the actual state at times of load changes. A nice overlapping between states has been observed. One may conclude that the EKF works properly under load conditions.

In figure (8), the standard deviation of measurement noise has been changed and the trace of error covariance matrix, Trace (P), is calculated in each time. Being the covariance matrix P is a measure of how we are certain in the measurements; one can expect that the trace of matrix will show large values for large values of standard deviation of measurement noise. This conclusion has been reported in Fig.(8).

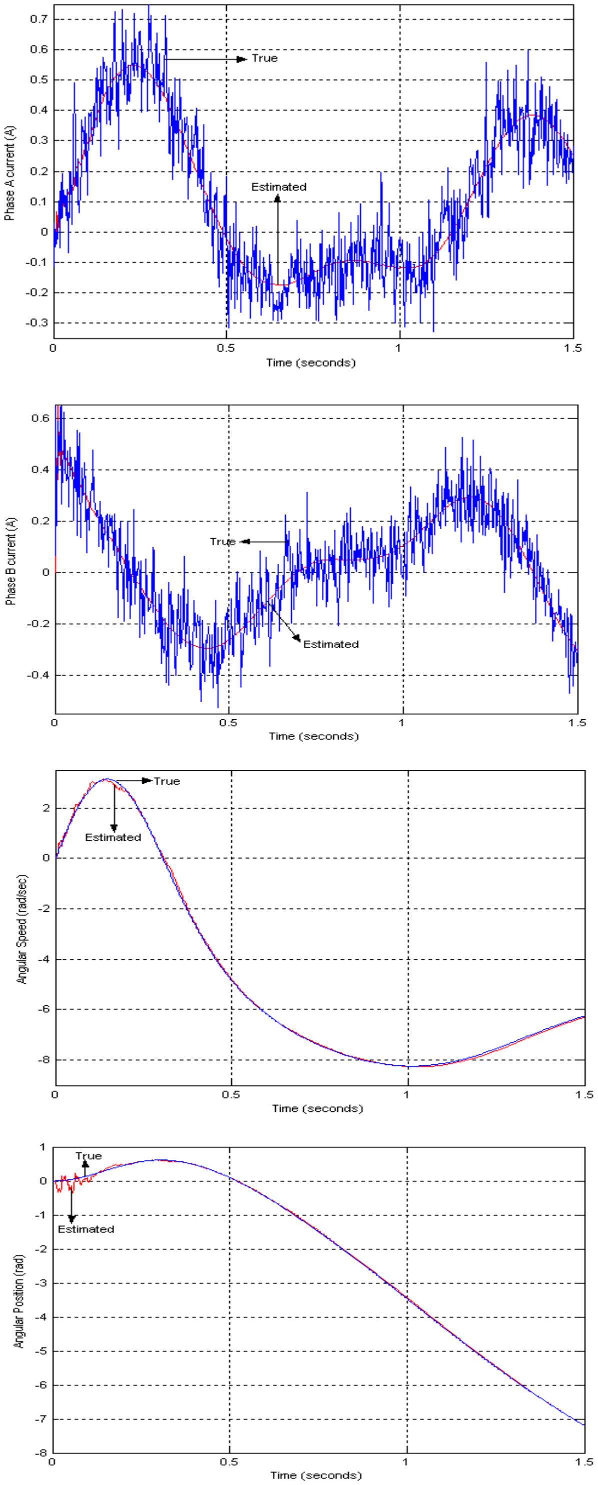

Figure (4) Estimated and actual states of the motor.

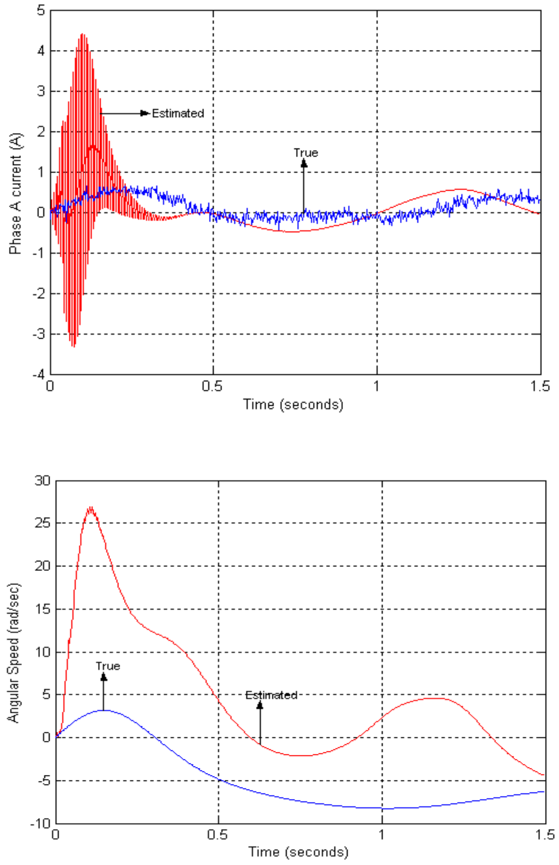

Figure (5) Estimated and actual states (current and speed) of the motor at sampling time

(T=2.93 ms)

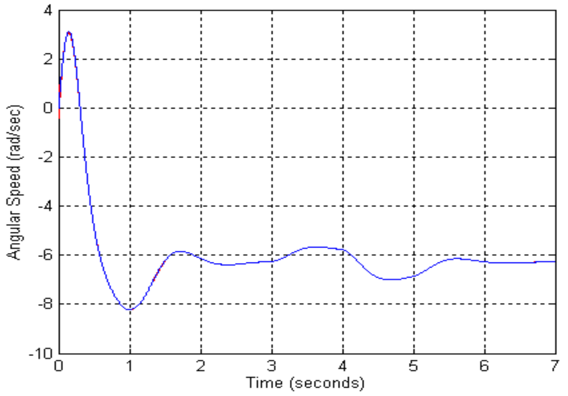

Figure (6) Estimated and actual states of the motor.

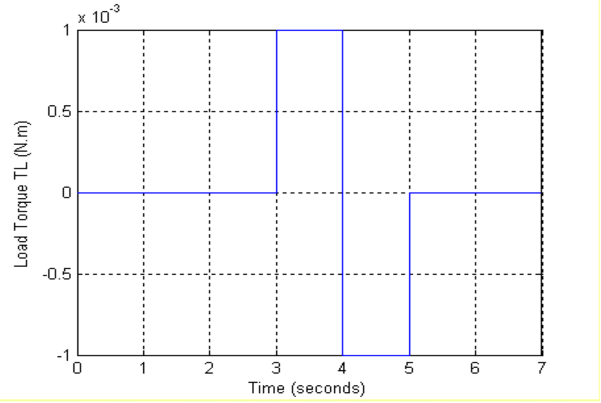

Figure (7) Change of Motor Torque Load

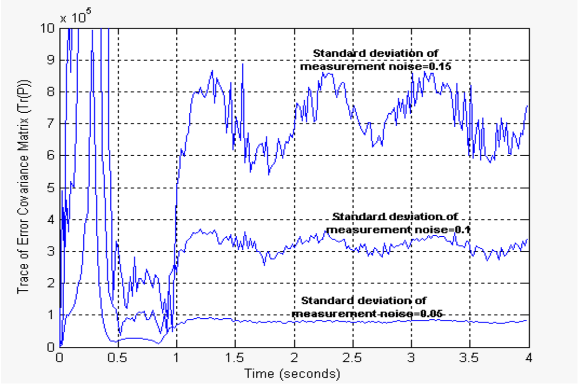

Figure (8) Trace of the error state covariance matrix with different standard deviations of measurement noises

Conclusion

Based on the observations of the simulated results one might highlights the following points:

Inspection of the figure (8) shows that if the measurement noise is large, so ***P*** will be large two and we don’t have much certainty about the measurement *y* when computing

. On the other hand, if the measurement noise is small, so ***P*** will be small and $\hat{x}$ the next

. $\hat{x}$ we will have a lot of certainty to the measurement when computing the next

- One can easily conclude that the EKF estimator could successively estimate the motor states and the estimator showed an excellent noise rejection capability.
- The application of Kalman Filter is restricted by the limitation of sampling period. A serious stability problem will arise as the sampling time is increased to a specified value. As the Kalman gain K suffers singularity at the increased sampling time.
- The EKF estimator shows good tracking performance in spite of load exertion during estimation process.

References

[1] B. Akin, “State Estimation Techniques for Speed Sensor less Field Oriented Control of Induction Machine”, Master's thesis, The Middle East technical university, 2003.

[2] B. K. Bose, “Modern Power Electronics and AC Drive”, University of Tennessee, Knoxville, Prentice Hall, 2002.

[3] D. Simon, “Optimal State Estimation: Kalman, H_∞, and Nonlinear Approaches,” John Wiley & Sons Inc, 2006.

[4] G. F. Franklin, “Digital Control of Dynamic Systems,” Addison-Wesly Longman, Inc, 1998.

[5] D. Simon, “Kalman Filtering” Embedded Systems Programming, JUNE, 2001.

[6] A. Gelb, “Applied Optimal Estimation,” Cambridge, Mass, 1974.

[7] R. G. Brown ,“Introduction to Random Signal Analysis and Kalman Filter,” Johon Wiley and Sons, Inc, 1983.

[8] D. Simon "Kalman Filtering and Neural Networks," Johon Wiley and Sons, Inc, 2001.

[9] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," ACM, Inc, 2001.

[10] M. S. Grewal, A. P. Andrews, "Kalman Filtering: Theory and Practice Using Matlab" John Wiley and Sons Inc, 2001.

Appendix I

The parameters of two-phase synchronous motor are listed in Table (1)

Table 1: Parameters of two-phase synchronous motor

Winding resistance	R_a	$2\,\Omega$
Winding inductance	L	3 mH
Motor flux constant	λ	0.1
ndard deviation of control input noises	σ_i	0.001 A
andard deviation of load torque noise	σ_T	0.05 $rad/\sec^2$
andard deviation of measurement noise	σ_m	0.1 A
Moment of Inertia	J	0.002
Frequency	f	1 Hz

Appendix II

Two-phase Motor S-function-Type *m*-File Listing

```
function [sys,x0]=SfunctionDynamicModel (t,x,u,flag,T)

global a11 a13 a31 a33 a43 b11;

global StateNoise MeasNoise;

if flag==0

Initial;

x0=zeros(4,1);     % zero Initial states

sys=[0,4,4,3,0,0];% zero continous states:four discrete states

  % four outputs and three inputs.

elseif flag==2

U=[u(1);u(2)];    % receive sinusoidal inputs.

Tl=u(3);              % receive Torque load input.

% X=A*X+B*U+M*TL+W  :  State equation

% Y=C*X+V                        :  Output equation

% where

%A=[(-Ra/L),0,(lambda/L)*sin(x(4)),0;

%   0,(-Ra/L),-(lambda/L)*cos(x(4)),0;

% -(3/2)*(lambda/J)*sin(x(4)),(3/2)*(lambda/J)*cos(x(4))-(F/J),0;

%   0,0,0,x(3)];

%   B=[1/L;1/L];

%   C=[1 0 0 0;0 1 0 0];

%   M=[0;0;0;-1/L]
```

```
W=StateNoise.*randn(4,1);
V=MeasNoise*randn(2,1);
% Ad=I+A*T+(1/2)*(A*T)^2+......
% Ad=[1-(Ra/L)*T,0,(lambda/L)*sin(x(4))*T,0;
%     0,1-(Ra/L)*T,-(lambda/L)*cos(x(4))*T,0;
%    -(3/2)*(lambda/J)*sin(x(4))*T,(3/2)*(lambda/J)*cos(x(4))*T,
%      1-(F/J)*T,0;
%       0,0,T,1];
Ad=[a11,0,a13*sin(x(4)),0;
    0,a11,-a13*cos(x(4)),0;
    a31*sin(x(4)),-a31*cos(x(4)),a33,0;
    0,0,a43,1];
% Bd=B*T : Wd=W*T : Cd=C  : D=0
Bd=[b11 0;0 b11;0 0;0 0];
% C=[1 0 0 0;0 1 0 0];
Wd=W*T;
Md=[0;0;0;-T/L];
% X=A*X+B*U+W
Xd=Ad*x+Bd*U+Wd+Md*TL;
% Y=CX+V
%Y=C*x+V
sys=[Xd];
elseif flag==3
```

```
 U=[u(1);u(2)];
 V=MeasNoise*randn(2,1);
 C=[1 0 0 0;0 1 0 0];
 Y=C*x+V;
 sys=[Y(1) Y(2) x(3) x(4)];
else
    sys=[];
end
```

Appendix III

State Estimator S-function-Type *m*-File Listing

```
function [sys,x0]=SfunctionStateEstimator (t,x,u,flag,T)
global a11 a13 a31 a33 a43 b11;
global Q R P;
if flag==0
Initial2;
x0=zeros(4,1);
sys=[0,4,4,5,0,0];
elseif flag==2
U=[u(1);u(2)]; % The quadrature sinusoidal inputs.
Y=[u(3);u(4)];  % The actual currents inputs.
TL=u(5);           % Reception of Torque Load.
xest=x;
% Compute the partial derivative matrices
```

```
dA=[a11,0,a13*sin(xest(4)),a13*xest(3)*cos(xest(4));
    0,a11,-a13*cos(xest(4)),a13*xest(3)*sin(xest(4));
   a31*sin(xest(4)),-a31*cos(xest(4)),a33,
   a31*(xest(1)*cos(xest(4))+xest(2)*sin(xest(4)));
        0,  0, a43, 1];
C=[1 0 0 0;0 1 0 0];
% Compute the Kalman gain
K=P*C'*inv(C*P*C'+R);
% Update the state estimate
Ad=[a11,0,a13*sin(x(4)),0;
    0,a11,-a13*cos(x(4)),0;
    a31*sin(x(4)),-a31*cos(x(4)),a33,0;
    0,0,a43,1];
Bd=[b11 0;0 b11;0 0;0 0];
Md=[0;0;0;-T/J];
Xd_est=Ad*xest+Bd*U+Md*TL+K*(Y-C*xest);
% Update the estimation error covariance
P=dA*((eye(4)-K*C)*P)*dA'+Q;
sys=Xd_est;
elseif flag==3
 sys=[x(1) x(2) x(3) x(4)];
else
    sys=[];
end
```