Dealing with Ill Conditioning in Recursive Parameter Estimation for a Synchronous Generator

by

Carlos Eduardo Niño Barón

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
ELECTRICAL ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS
2006

Approved by:

_____	_____
Agustin Irizarry Rivera, Ph.D. Member, Graduate Committee	Date
_____	_____
Efrain O'Neill-Carrillo, Ph.D. Member, Graduate Committee	Date
_____	_____
Miguel Velez-Reyes, Ph.D. President, Graduate Committee	Date
_____	_____
José A. Colucci, PhD Representative of Graduate Studies	Date
_____	_____
Isidoro Couvertier, PhD. Chairperson of the Department	Date

ABSTRACT

This thesis presents how to deal with ill-conditioning in recursive parameter estimation for a synchronous generator using subset selection, the Extended Kalman Filter (EKF), and the Iterated Extended Kalman Filter (IEKF). We present how the quality of the estimates in ill-conditioned parameter estimation problems is significantly affected by noise and how by proper modifications to the EKF, we still extract useful parameter estimates from low quality data. The modifications to the EKF and IEKF are based on the subset selection method, where only a subset of parameters is estimated from the available data and the other parameters are fixed to prior values. The reduced order parameter estimation problem is better conditioned allowing the extraction of good estimates from the available data. Simulation studies on the identification of a linearized model of a synchronous generator are used to illustrate the concepts being studied in this work. Simulation results show how the modifications to the EKF and IEKF based on the subset selection method result in convergent algorithms when their application to the original full problem was not. We also show that for this case the additional computational effort needed for the IEKF does not result in significant improvement in the quality of the estimates over those obtained with EKF.

RESUMEN

Esta tesis presenta los algoritmos de Filtro Extendido de Kalman (EKF), Filtro Iterado Extendido de Kalman (IEKF) usando subset selección para el manejo de mal acondicionamiento en identificación recursiva de parámetros de generadores sincrónicos. También es presentado como el EKF puede ser modificado para extraer información suficiente para calcular parámetros a partir de datos de baja calidad. La metodología propuesta se fundamenta en la sección de parámetros, donde un grupo de parámetros es fijado antes de realizar el proceso de estimación para reducir el mal acondicionamiento. Para mostrar los conceptos propuestos en este trabajo, fueron realizadas simulaciones empleando modelos linealizados de un generador sincrónico. Los resultados simulados muestran que el EKF de orden completo no converge bajo condiciones de ruido, sin embargo cuando se emplea la metodología de "subset selection" se pueden estimar 7 de 9 parámetros de forma confiable.

ACKNOWLEDGMENT

I would like to thank Dr. Miguel Vélez-Reyes for giving me the opportunity to have this research experience and providing assistance and support to present this work.

I would also want to thanks to the professors in my committee: Dr. Agustín Irizarry Rivera and Efrain O'Neill Carrillo for their comments and suggestions in this thesis, thanks for their time to revise and evaluate the final document as well. I want to acknowledgments go to my friends Lola, Ana, Jorge and the DSP family. I would also want to thank my parents José Armando Niño and Maria Del Carmen Barón and my sister Monica Andrea and especially thanks go to my girlfriend Solimar, for her love and support during mi Master.

This work was primarily supported by the National Science Foundation under grant EEC- 0328200. This work made use of ERC Shared Facilities supported by the National Science Foundation under Award Number EEC-9731677.

CONTENTS

LIST OF TABLES VII

LIST OF FIGURES VIII

CHAPTER 1 INTRODUCTION 1

1.1 JUSTIFICATION AND OBJECTIVES 1
1.2 OBJECTIVES 2
1.3 LITERATURE REVIEW 3
1.3.1 PARAMETER IDENTIFICATION: BATCH 3
1.3.2 RECURSIVE PARAMETER IDENTIFICATION 4
1.3.3 RECURSIVE ESTIMATION IN SYNCHRONOUS GENERATORS 5
1.4 STANDARD TESTS FOR PARAMETER DETERMINATION IN SYNCHRONOUS GENERATORS 6
1.4.1 TEST FOR DIRECT AXIS COMPONENTS DETERMINATION 6
1.4.2 TEST FOR QUADRATURE AXIS PARAMETERS 12
1.5 CONDITIONING ANALYSIS 14
1.6 THESIS OVERVIEW 16

CHAPTER 2 SYNCHRONOUS GENERATOR MODEL 17

2.1 STATOR EQUATIONS 18
2.2 ROTOR EQUATIONS 19
2.3 MECHANICAL MODEL 20
2.4 DQ0 TRANSFORMATION 20
2.5 INCREMENTAL GENERATOR MODEL 22

CHAPTER 3 BATCH PARAMETER ESTIMATION 25

3.1 INTRODUCTION 26
3.2 LINEAR LEAST SQUARES 27

3.3 GAUSS-NEWTON METHOD. 28
3.4 LEVENBERG-MARQUARDT METHOD 33

CHAPTER 4 EXTENDED KALMAN FILTER 37

4.1 ALGORITHM DESCRIPTION 37
4.2 EXTENDED KALMAN FILTER APPLIED TO SYNCHRONOUS GENERATOR MODEL 45
4.3 CONDITIONING ANALYSIS 49

CHAPTER 5 MODIFIED EXTENDED KALMAN FILTER 53

5.1 SUBSET SELECTION 53
5.2 ITERATED EXTENDED KALMAN FILTER (IEKF) 62

CHAPTER 6 CONCLUSIONS AND RECOMMENDATION 69

6.1 CONCLUSIONS 69
6.2 FUTURE WORK 69

REFERENCES 72

APPENDIX A 75

APPENDIX B 79

APPENDIX C 85

LIST OF TABLES

Table 1 Nominal parameters of synchronous generator 29

Table 2 Steady state results of Estimation using the EKF: noise free and noisy analysis 48

Table 3 Hessian matrix of EKF (noisy case) 50

Table 4 Hessian's matrix eigenvalues..... 51

Table 5 Condition Number of Subset selection batch technique..... 54

Table 6 Steady state Results of Subset Selection applied to Gauss Newton method 55

Table 7 Steady state results from estimation using the subset selection- EKF..... 61

Table 8 Steady state results of Estimation using the subset selection- IEKF 68

Table 9 comparative analysis of the EKF and the UD-EKF..... 84

Table 10 EKF code files 85

Table 11 IEKF and sub set selection code files 101

LIST OF FIGURES

Figure 1 Analysis of a.c. component of short-circuit component [IEEE Standard, 1995]. 8

Figure 2 Field winding circuits for direct axis transient open circuit time constant..... 10

Figure 3 Voltage recovery test- short circuit field. 11

Figure 4 Method for determining the sub transient quadrature reactance 12

Figure 5 Synchronous generator schematic diagram 17

Figure 6 DQ mechanical references..... 21

Figure 7 DQ0 reference 21

Figure 8 General parameter identification structures..... 26

Figure 9. Input to the Gauss 100 MVA synchronous generator 29

Figure 10. Output to the 100 MVA synchronous generator 30

Figure 11. Input to the 100 MVA synchronous generator (noisy case)..... 30

Figure 12. Output to the 100 MVA synchronous generator (noisy case) 31

Figure 13 Estimated Parameters applying Gauss-Newton (noise free case) 32

Figure 14 Estimated Parameters applying Gauss-Newton (noisy case) 33

Figure 15 Estimated Parameters applying Levenberg-Marquardt (noise free case)..... 35

Figure 16 Estimated Parameters applying Levenberg-Marquardt (noisy case)..... 36

Figure 17. Extended Kalman Filter Flowchart 44

Figure 18. Estimated States EKF full order (noise free case)..... 45

Figure 19. Estimated output EKF full order (noise free case) 46

Figure 20. Estimated parameters EKF full order (noise free case)..... 46

Figure 21. Estimated States EKF full order (noisy case) 47

Figure 22. Estimated output EKF full order (noisy case) 47

Figure 23. Estimated parameters EKF full order (noise case) 48

Figure 24. Jacobian of Synchronous Generator Model 50

Figure 25. Condition number of error covariance matrix (noise free analysis) 52

Figure 26. Condition number of error covariance matrix (noisy analysis) 52

Figure 27. Estimated sates with Xd and K fixed for subset selection EKF 56

Figure 28. Estimated parameters with Xd and K fixed for subset selection EKF 56

Figure 29. Estimated sates with Td0' and K fixed for subset selection EKF 57

Figure 30. Estimated parameters with Td0' and K fixed for subset selection EKF 57

Figure 31. Estimated sates with K fixed SS-EKF (noise free) 58

Figure 32. Estimated outputs with K fixed SS-EKF (noise free) 59

Figure 33. Estimated parameters with K fixed SS-EKF (noise free) 59

Figure 34. Estimated states with Xd", and K fixed SS-EKF (noisy case) 60

Figure 35. Estimated outputs with Xd" and K fixed SS-EKF (noisy case) 60

Figure 36. Estimated parameters with Xd" and K fixed SS-EKF (noisy case) 61

Figure 37. Iterated Extended Kalman Filter algorithm 64

Figure 38. Estimated states with K fixed SS-IEKF (noise free) 65

Figure 39. Estimated outputs with K fixed SS-IEKF (noise free) 65

Figure 40. Estimated parameters with K fixed SS-IEKF (noise free) 66

Figure 41. Estimated states with Xd" and K fixed SS-IEKF (noisy case) 66

Figure 42. Estimated outputs with Xd", Tq0" and K fixed SS-IEKF (noisy case) 67

Figure 43. Estimated parameters with Xd" and K fixed SS-IEKF (noisy case) 67

Figure 44 UD factorization algorithm..... 81

Figure 45 UD factorization (P correction and K calculus) 82

Figure 46 UD factorization (P predictor)..... 83

CHAPTER 1 INTRODUCTION

This chapter presents the concepts which justify the present research. The principal objective proposed to solve the problem is presented and a literature review is made to show the state of art of the parameter identification problem.

1.1 JUSTIFICATION AND OBJECTIVES

The synchronous generator is the principal component of power systems and the fundamental element used to control their dynamical stability. For this reason, it is very important to have accurate models of these machines to analyze and to predict its transient response and its influence in the power system. Dynamic models for synchronous generators derived from basic principles of physics often contain parameters whose values cannot be accurately predicted theoretically and are estimated from measured data. Traditional methods for parameter identification are presented in the [IEEE Standard, 1995]. These methodologies assume a well-know structure of the generator and the possibility to make tests, which are impossible in many cases.

Furthermore, the characteristics of the generator change during its operational life and depend on operating conditions so it is desirable to be able to monitor these parameters during operation of the machine. Restrictions in the type of inputs that can be applied to synchronous generator during operation might result in data that does not contain rich information about all model parameters. Hence, we can not accurately estimate all model parameters using the available data. Mathematically this result in a parameter estimation problem that is ill posed or ill conditioned [Burth et al 1999, Jauregui 2001 and Velez 1992]. Ill-posedness implies that the parameters are not identifiable from the data. Ill conditioning implies that the computed estimates are very sensitive to perturbations in the input data due to sensor noise or the computation process and therefore ill-conditioning might render results from the estimation algorithm in many instances useless even though the model is physically sound and identifiable.

In [Burth et al 1999, Jauregui 2001], a method was developed to deal with ill conditioning in batch parameter estimation for a synchronous generator. To deal with ill conditioning, it was proposed that a subset of parameters of the generator model is fixed to prior values while estimating the remaining parameters from the available data. This is the so called subset selection method (or parameter selection [Polak, 2001]) method.

Prior information about the fixed parameters can come from manufacturer's data or from IEEE standard tests. Results in [Burth et al 1999, Jauregui 2001] showed that the reduced order problem is significantly less sensitive than the full order parameter estimation problem, and algorithms for parameter estimation in the reduced order problem have significant better convergence behavior than their counterparts applied to the full order problem.

In this work, we are investigating the extension of the batch subset selection method to the recursive case and what additional degradation is introduced by the sequential processing when dealing with the ill-conditioned parameter estimation problem for the synchronous generator. The Extended Kalman Filter (EKF) and the Iterated EKF (IEKF) are compared for this problem.

1.2 OBJECTIVES

The main objective of this research was the development of robust recursive parameter estimation algorithms for the synchronous generator. To achieve the main objective, we need to achieve the following secondary objectives:

- Develop a sensitivity analysis to identify which parameters increase the ill conditioning problem and which components are affected by the noise.

- Implement the Iterated Extended Kalman Filter for parameter identification applied to the synchronous generator.

- Implement the IEKF with the subset selection, study the convergence and improvements obtained by the mixing of these two methodologies.

1.3 LITERATURE REVIEW

A literature review was made to have into account the most recently advances in the areas of this research work. Most of the works have a common motivation that is try to resolve the problem of the parameter identification.

There are various researches in the area of parameter identification; the first part of literature-reviewed shows the concepts related to batch parameter identification, the second part presents the concepts in recursive parameter identification and the third part presents the application of recursive identification to synchronous generator.

1.3.1 PARAMETER IDENTIFICATION: BATCH

In [Velez-Reyes et al, 1992] the concepts about the ill-conditioned parameter identification are presented. This thesis present the errors and the non-convergence problems generated by ill conditioning parameter estimation, they analyze, and propose the decomposed methods for fast convergence.

In [Jáuregui, 2001] the analysis and mitigation of ill-conditioning problem in parameter identification for a synchronous generator is presented. That thesis analyzed the conditioning of the parameter estimation problem for a linearized model of the synchronous generator developed in [Burth et al, 1999] using local sensitivity analysis; present robust methods for parameter estimation using subset selection, Tikhonov, and truncated singular value decomposition regularization. Results show the effectiveness of local sensitivity analysis to predict ill-conditioning and the effectiveness of regularization methods to deal with ill-conditioned parameter estimation.

In [Polak, 2000] two techniques for improvements in ill-conditioned parameter identification; parameter selection and ridge regression are presented. These two methodologies were combined in a new estimator: the Regularized Estimation of Selected Parameters. The advantages of this method are: simplification of the identification algorithm, improvement in the numerical condition of identification, and enhancement in the accuracy of estimated parameters. The Regularized Estimation of Selected Parameters was applied to a simulation model were the error analysis shows its improvements compared with the traditional methods. This method is related to subset selection proposed of [Velez, 1992] but they do not know how to identify which parameters to fix and how many.

In [Huaman, 2005] is presented an algorithm based on a modification of the variable dimension Newton-Raphson method presented in [Ng et al, 2000]; this is applied for the same linearized model as in [Jauregui, 2001] and [Burth, 1999]. The methodology of [Ng et al, 2000] to solve ill conditioned nonlinear systems of equations is modified to deal with nonlinear least squares problems. The approach is applied to data from the FC5HP synchronous generator located at the Four Corners Generating Station of the Arizona Public Service Company (APS), rated at 483 MVA. The presented results show the improvement in the convergence of the algorithm against the subset selection and the Tikhonov implemented in [Jáuregui, 2001].

1.3.2 RECURSIVE PARAMETER IDENTIFICATION

In [Vahidi et al, 2003] is proposed recursive least squares with multiple forgetting factors and apply these methodology to the estimation of vehicles mass. It presents a briefly discussion of the recursive least square scheme for systems with time varying parameters. The paper also analyses the problem to uses a constant forgetting factor and proposes a recursive least squares with different forgetting factors. Although, they could not prove the algorithm convergence, nor define a region of convergence for the algorithm, they demonstrated, with both simulated and experimental data that

incorporating two distinct forgetting factors the effectiveness resolving the difficulties in estimating mass and time-varying grade. The experimental setup and particular issues with experimental data were also discussed.

[Ngia, 2000] presents a methodology to improve the Recursive Least Squares by the separation of linear and nonlinear parameters in a Hammerstein model. The approach focuses in the solution in the nonlinear part. The modification has the same computational loads as the algorithms that are resulted from the original unseparated problem, but it converges faster and reduces tracking problems. In a system identification example, the proposed algorithms are shown to have better convergence and tracking properties than alternative algorithms.

[Gunnarsson, 1996] presents the problems generated by recursive identification like Recursive least squares, where the gains (covariance matrix) tend to zero in steady state. One approach to avoid that is the use of an exponential forgetting factor; however that approach made the system sensitive to noise when it has poor excitation. This reason suggest the uses of regularization techniques that handles that problem. The regularization improves convergence performance for the estimator.

[Telford et al, 2003] present a recursive parameter identification applied to an induction machine. That paper shows the importance of the accurate calculation of electrical and mechanical parameter in the control vector, the solution presented is the application of recursive parameter identification, using Recursive least squares and least mean squares. Simulation and experimental results are presented using SIMULINK, which demonstrate the effectiveness of the online parameter identification and control loop tuning technique.

1.3.3 RECURSIVE ESTIMATION IN SYNCHRONOUS GENERATORS

[Abido et al, 1997] propose an On-line identification of synchronous machines using radial Basis Function Neural Network. The advantage of this method is the capability of the estimator to capture the nonlinear operating characteristics of the synchronous machine. The result of the proposed identifier performance due to square

and uniformly distributed random variations in both mechanical torque and field voltage are compared with that obtained by time domain simulations. Correlation- based model validity test using residuals were done to exanimate the validity of the proposed identifier.

[Wang et al, 1995] presents a technique suitable for identifying several typical machine saturation models. It is shown that the technique can be used to identify nonlinearities associated with saturation and cross-magnetizing effects. An algorithm based on the weighted least mean square was designed to estimate all parameters of the machine model and the associated saturation functions. Experimental studies using online measured data of the Tai-power system recorded by the plant transient recording and analysis system are included.

1.4 STANDARD TESTS FOR PARAMETER DETERMINATION IN SYNCHRONOUS GENERATORS

There are various synchronous generator models [Kundur, 1996] which commonly has the parameters presented in the [IEEE, Standard 1991]. The determination of these parameters follows the standard test presented in [IEEE Standard, 1995]. Based on the classical transformation DQ, these parameters are divided in two groups: direct axis components and the quadrature axis components. The tests performed to determine the synchronous generator parameters are presented below.

1.4.1 TEST FOR DIRECT AXIS COMPONENTS DETERMINATION

There are four different tests performed to determine the direct axis elements of synchronous generator: slip test, short circuit armature test, field short circuit, and the armature recovery test. The slip test presented is performed by driving the rotor to a speed very closed to the synchronous speed (but not equal), with the field winding open circuited and the stator connected to an external source with nominal voltage and

frequency; the armature and current are measured to calculate the saturated direct reactance defined by:

$$X_{ds} = \frac{E_{\max}}{I_{\min}}(p.u.)\, a \quad certain \quad saturated \quad value \qquad (1.1)$$

where E and I corresponding to values of the measured signals.

In the short circuit test the transient and subtransient reactances are calculated, in this test a short circuit is applied to the stator winding of the synchronous generator. The current is described by the equation:

$$i(t) = \frac{E}{X_{ds}} + \left(\frac{E}{X'_d} - \frac{E}{X_{ds}}\right)e^{\frac{-t}{T'd}} + \left(\frac{E}{X''_d} - \frac{E}{X'_d}\right)e^{\frac{-t}{T''d}} \qquad (1.2)$$

where

i is the a.c, rms short-circuit current p.u.

E is the a.c. r.m.s voltage before the short circuit p.u.

t is the time in seconds measured from the instant of the short circuit

X_{ds} is the saturated direct axis reactance p.u

X'_d is the transient direct axis reactance p.u

X''_d is the sub transient direct axis reactance p.u

T'_d is the transient constant time

T''_d is the sub transient constant time

In this equation, it is assumed that the current is composed of a constant term and two decaying exponential terms, (the third decays faster than the second). In this test the synchronous generator is operated to nominal voltage and speed and then a short circuit is applied in the armature, the currents are measured. The current's constant value is subtracted and the result is plotted in semi logarithmic scale as a time function, (Figure 1,

curve B) the curve would be a straight line and then appears another component that rapidly decays to zero. The fast component corresponds to the sub transient component and the other is the transient component. A typical figure is presented in Figure 1:

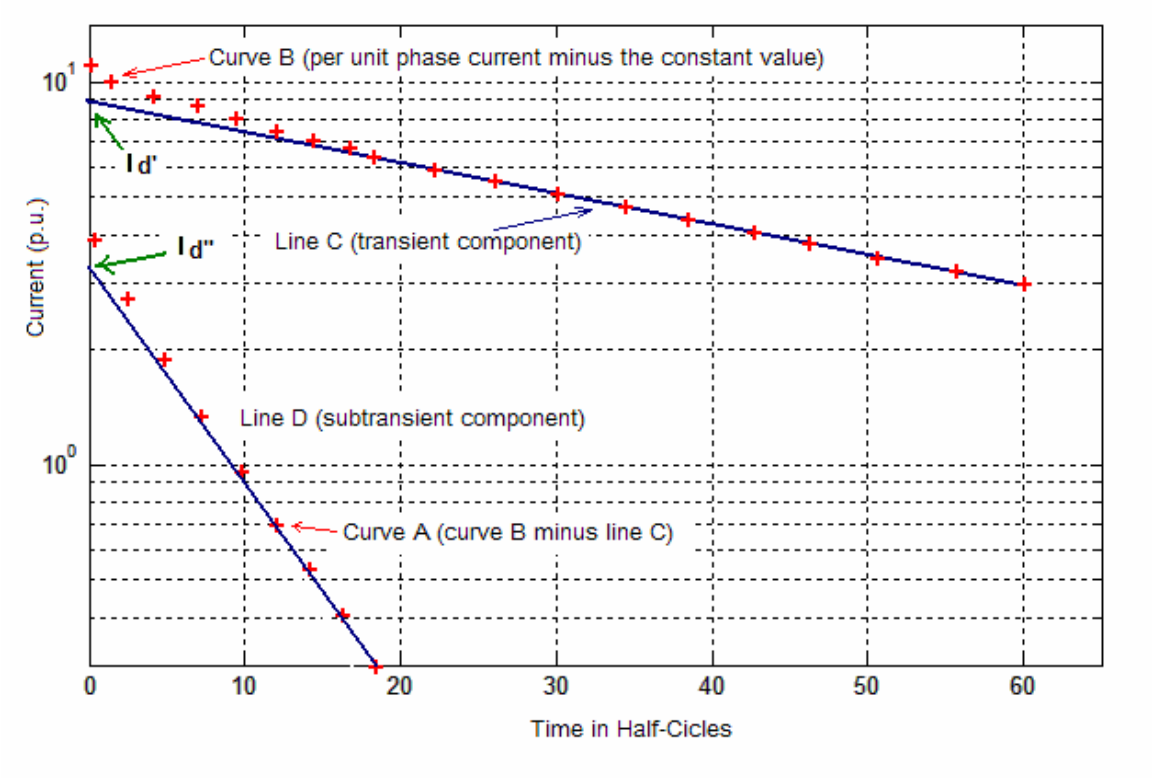

Figure 1 Analysis of a.c. component of short-circuit component [IEEE Standard, 1995].

The direct-axis transient reactance Xd' is equal to the ratio of the open-circuit voltage to the value of the armature current obtained by the extrapolation of the envelope of the ac component of the armature current wave to the time of application of the short circuit (neglecting the rapid variation of current during the first few cycles). Figure 1 illustrates this method to calculate the direct-axis transient reactance.

The values of the three transient currents are averaged and this value is used to calculate the transient reactance by:

$$X'_d = \frac{E}{I'} \qquad (1.3)$$

The direct-axis subtransient reactance Xd", is determined from the same test used for the transient reactance. For each phase, the values of the difference between the ordinates of curve B and the transient component (line C), are plotted as curve A (on the same sheet) to give the subtransient component of the short-circuit current as shown in Figure 1.

The result is expected to be very a straight line on the semilogarithmic plot. Extending the straight line (line D) back to time zero gives the initial value of the subtransient component of the short-circuit current. Preference in locating line D should be given to the first few points, corresponding to the first few cycles after application of the short circuit, as they are normally the points that can be determined most accurately. The sum of the initial subtransient component, the initial transient component, and the sustained component for each phase gives the corresponding value of I".

The procedure described above requires that the currents are weighed. If (a) is the largest value and (c) is the smaller, the weighed average computed as follows:

$$I''(1) = \sqrt{\frac{2}{3}\left(a^2 + b^2 - ab\right)} \qquad (1.4)$$

$$I''(2) = \sqrt{\frac{2}{3}\left(a^2 + c^2 - ac\right)} \qquad (1.5)$$

$$I''(3) = \sqrt{\frac{2}{3}\left(b^2 + c^2 - bc\right)} \qquad (1.6)$$

Finally the three components are averaged by:

$$I''(1) = \frac{3I''(1) + 2I''(1) + I''(1)}{6} \qquad (1.7)$$

the sub-transient current is used to calculate the sub-transient reactance:

$$X''_d = \frac{E}{I''} \qquad (1.8)$$

The third test is the field short circuit. This test requires operating the generator to the rated voltage and speed. The field is connected to the exciter source as presented in Figure 2. To perform the test is necessary to close field discharge contact and then open the contacts of the exciter (the series resistor is used to protect the exciter).

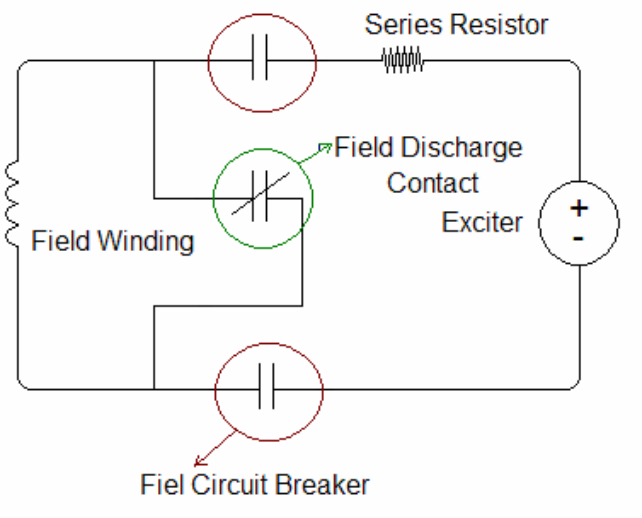

Figure 2 Field winding circuits for direct axis transient open circuit time constant.

The test is performed with the armature winding open and the machine operated at rated speed. The r.m.s. residual armature voltage is subtracted from the r.m.s. armature voltage obtained from the oscillogram at selected points of time. The resulting varying component of voltage is plotted against time on semi logarithmic paper with the armature voltage on the logarithmic scale (the same process for the Figure 1). Normally, the curve is approximately a straight line if the few initial points of rapid decay are neglected. Extrapolation of the curve, neglecting the first few cycles, back to the moment of closing of the field-discharge contact, gives the effective initial voltage. The time in seconds for

the armature voltage to decay to 0.368 times the effective initial voltage is the open-circuit transient time constant T'_{d0}.

The last test is armature voltage recovery test , when the machine is running at rated speed and with a field connected to a DC source (the same voltage required to have the nominal armature voltage with no load), the line to line armature voltage is measured following the sudden opening a steady state short circuit in the armature. The differential voltage (AE) is obtained at frequent intervals by subtracting the average of the three rms voltages (obtained from the oscillograms) from the average of the three rms steady-state voltages. The subtransient voltage (curve A of Figure 3) is obtained by subtracting the transient component of differential voltage (line C) from the differential voltage (curve B). A straight line (line D) of Figure 3) is fitted to this plot, giving preference to the earliest points if they do not follow a linear trend. The direct-axis subtransient open-circuit time constant is the time, in seconds, on the straight line corresponding to 1 / e or 0.368, times the ordinate of the line at the instant of opening the circuit T''_{d0}.

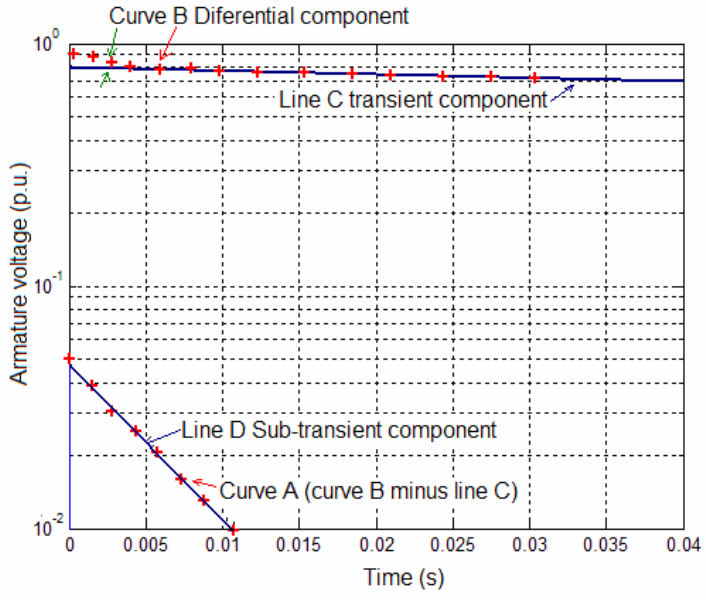

Figure 3 Voltage recovery test- short circuit field.

1.4.2 TEST FOR QUADRATURE AXIS PARAMETERS

The quadrature components are calculated by the slip test presented below. The quadrature-axis reactance is calculated by the equation:

$$X_{qs} = \frac{E_{\min}}{I_{\max}}(p.u.)\, a \quad certain \quad saturated \quad value \tag{1.9}$$

where E and I correspond to the value of voltages and currents from the slip test (described previously).

The stationary test is used to calculate the subtransient quadrature-axis reactance. For this test the rotor is stationary and the field winding is short-circuited. A single phase reduced voltage at nominal frequency is applied to two armature terminals; the connection is shown in Figure 4. The constant X is calculated using the measured values.

$$X = \frac{E}{I} \tag{1.10}$$

where

E is the applied line to line voltage, in p.u. of base line to neutral voltage

I is the line current in p.u. of the base line current

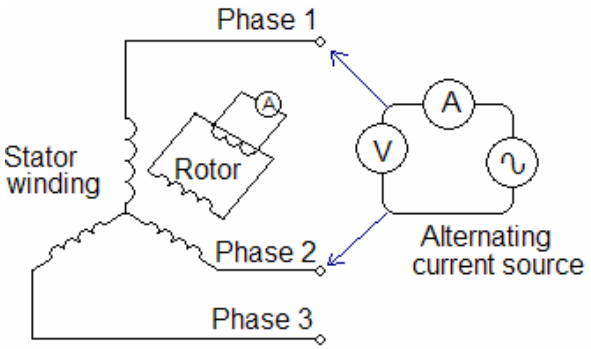

Figure 4 Method for determining the sub transient quadrature reactance

With the same applied voltage an equal procedure is used for the remaining connections to calculate other Y and Z (phases 2-3 and phases 1-3). One important consideration is that the rotor must be locked in a fixed position. The values of X, Y and

Z correspond to the three values of the stationary impedance and every one differ from the other by 120 electrical degrees. Using these impedances the constants K and M are calculated as:

$$K = \frac{X + Y + Z}{3} \tag{1.11}$$

$$M = \sqrt{(Y - K)^2 + \frac{(Z - X)^2}{3}} \tag{1.12}$$

and the subtransient reactance is calculated by

$$X''_q = \frac{K \pm M}{2}(p.u.) \tag{1.13}$$

Usually the reactance corresponds to the maximum possible value of (1.13) and the plus sign is taken; however for some cylindrical rotor machines it corresponds to the minimum.

To determine the quadrature-axis subtransient constant time Tq0" the low armature voltage test is used. In this test, the machine is running asynchronously unloaded, this test is performed with the armature (primary) winding connected to a rated frequency symmetrical three-phase low-voltage supply to avoid influence of saturation, and the excitation winding is short-circuited. By positioning sensors on the shaft the direct and quadrature axis are obtained, they are used to determine the direct- and quadrature-axis components of voltage and current. The positioning signals are set to correspond to the quadrature (magnetic axis) by means of a line-to-line voltage at no load.

The transmitter in the shaft producing the positioning signal should be displaced around the shaft until its signal coincides with the instant when the line-to-line voltage E_{12} passes through zero. At this position the signal will coincide with the maximum of phase voltage E_3 which, at no load, corresponds to a q-axis voltage. When the machine is electrically loaded, the instantaneous values of the phase voltage E_3 and of the phase

current I_3 at the instant of the signal coincide with the q-axis components of this voltage and current.

After adjusting the recorders, the test measurements are performed. When the machine has reached an internal angle approaching 90±20 electrical degrees, an oscillogram is taken of the line to line voltage E_{12}, the phase current $I_{3,}$ and the q-axis signal, showing a time interval that includes switching off. Before disconnecting the applied voltage, the current I_q (0) and the voltage $E_d(0)$ are measured. The sequence voltages after switching off are determined from an osillogram. The voltages are plotted in a logarithmic scale against the time and the initial transient and subtransient components are found in a similar fashion as direct axis components.

The transient quadrature constant time T'_{q0} is the time required for the slow changing component of the open circuit armature winding voltage due to the quadrature-axis flux, following a sudden change in operating conditions, to decrease to 1 /e or 0.368 of its initial value.

The subtransient quadrature constant time T''_{q0} is the time required for the rapidly changing component present during the first few cycles in the open circuit armature winding voltage due to the quadrature-axis flux, to decrease to 1 /e or 0.368 of its initial value.

1.5 CONDITIONING ANALYSIS

The objective of conditioning analysis of the parameter estimation problem developed by [Vélez-Reyes,1992] is to study the sensitivity to a mapping between the data and the estimated parameters:

$$\hat{\alpha} = f(Y) \qquad (1.14)$$

This mapping is defined by an optimization problem. Depending on the optimization criterion and the model structure this mapping can be defined explicitly or implicitly. The technique used in this thesis to analyze the conditioning of parameter

estimates is based on local sensitivity analysis using relative condition numbers. The relative condition number measures the change in the parameter estimate caused by variations in the measured data, it is given by

$$\frac{|\Delta\alpha_i|}{|\alpha_i|} = S_y^{\hat{\alpha}_i} \frac{\|\Delta y\|}{\|y\|} \qquad (1.15)$$

where

$$S_y^{\hat{\alpha}_i} = \frac{\|y\|\|\nabla f_i\|}{\|\hat{\alpha}\|} \frac{|\hat{\alpha}|}{|\alpha_i|} \qquad (1.16)$$

is the sensitivity function, ∇f is the gradient of the i th component of f, and $\|\cdot\|$ is the Euclidean 2-Norm. A system is said to be singular if the condition number is infinite, and ill-conditioned if it is too large, where "too large" means roughly $\log(C) \geq$ the precision of matrix entries. We can identify two components in (1.16) which point out to potential sources of large values for the condition number. The first component is the colinearity factor

$$CF_i = \frac{\|y\|\|\nabla f_i\|}{\|\hat{\alpha}\|} \qquad (1.17)$$

which is associated to the conditioning of the Jacobian of the parameter estimation mapping and a scaling factor

$$SF_i = \frac{\|\hat{\alpha}\|}{|\alpha_i|} \qquad (1.18)$$

This later one is associated with the relative scaling of the i th parameter to the remaining parameters. We consider the colinearity factor to be the most important factor in analyzing parameter conditioning. The ill-conditioning of the parameter estimation problem has the detrimental effect of increasing the sensitivity of parameter estimates to noise and numerical truncation error, and to slow down the convergence of parameter estimation algorithms (batch and recursive) [Vélez-Reyes,1992]. In [Jauregui, 2001] the

ill conditioning causes a un-convergence of the method in the noisy simulations and also the unstability of the algorithm [Jingmin et al , 2001].

1.6 THESIS OVERVIEW

This thesis is organized as follows. Chapter 2 presents the electrical model used for parameter identification. Chapter 3 presents the concepts of parameter identification in synchronous generator by batch estimation methodologies. Chapter 4 presents the Extended Kalman Filter (EKF). Sensitivity analysis is presented to identify the ill conditioned parameters and the noise influence in the filter. The chapter 5 presents the variants to the EKF to reduce the ill conditioning problem; the Subset selection methodology and the Iterated Extended Kalman Filter (IEKF). These methodologies are studied for the noise free and the noisy signals and also for the noisy signals. The conclusions and recommendations of this thesis are presented in the Chapter 6. The appendixes present the UD factorization, the recursive least squares and the MATLAB files showing the implementations of the differents algorithms.

CHAPTER 2 SYNCHRONOUS GENERATOR MODEL

Modeling of synchronous generators is an important field of study in power systems. The book [Kundur, 1994] present several models used in power systems studies and many other references can be cited that present models of the synchronous generator of differing detail and purpose. In this study, we are primarily interested in the ill conditioning aspects and how to handle them in recursive identification so we pick a particular model that has been used in similar studies and do not attempt to compare with other models presented in the literature

A schematic diagram of the model used in stability studies [Kundur, 1994] is present the below

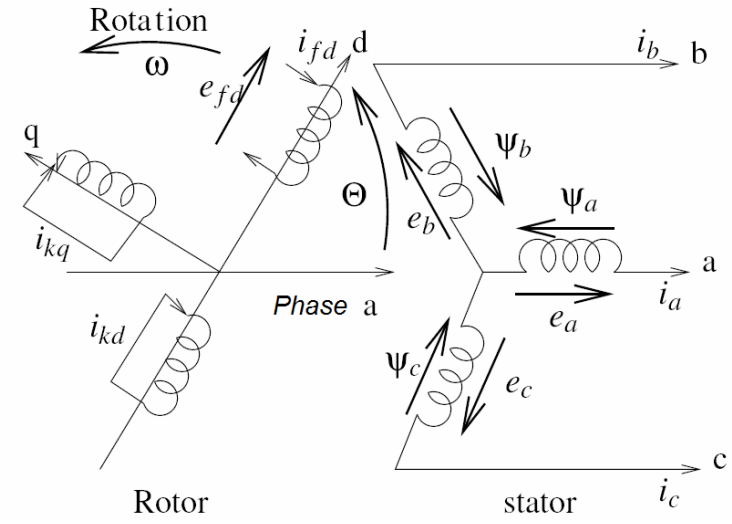

Figure 5 Synchronous generator schematic diagram

The equations which model the synchronous generator are divided in three principal parts: stator equations, rotor equations and mechanical model.

2.1 STATOR EQUATIONS

The electric model of the rotor is described by the equations (2.1) to (2.4).

$$\begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} = \begin{bmatrix} -R_a & 0 & 0 \\ 0 & -R_a & 0 \\ 0 & 0 & -R_a \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} \lambda_a \\ \lambda_b \\ \lambda_c \end{bmatrix} \tag{2.1}$$

where i_a, i_b, i_c, are the stator currents and $\lambda_a, \lambda_b, \lambda_c$ are the a, b, c flux linkages defined as follows:

$$\begin{aligned} \lambda_a &= -i_a[L_{aa0} + L_{aa2}\cos 2\theta] + i_b[L_{ab0} + L_{aa2}\cos(2\theta + \frac{2\pi}{3})] \\ &+ i_c[L_{ab0} + L_{aa2}\cos(2\theta - \frac{2\pi}{3})] + i_{fd}L_{afd}\cos\theta + i_{kd}L_{akd}\cos\theta - i_{kq}L_{akq}\sin\theta \end{aligned} \tag{2.2}$$

$$\begin{aligned} \lambda_b &= i_a[L_{ab0} + L_{aa2}\cos(2\theta + \frac{2\pi}{3})] - i_b[L_{aa2} + L_{aa2}\cos(2\theta - \frac{2\pi}{3})] \\ &+ i_c[L_{ab0} + L_{aa2}\cos(2\theta - \pi)] + i_{fd}L_{afd}\cos(\theta - \frac{2\pi}{3}) + i_{kd}L_{akd}(\theta - \frac{2\pi}{3}) \\ &- i_{kq}L_{akq}\sin(\theta - \frac{2\pi}{3}) \end{aligned} \tag{2.3}$$

$$\begin{aligned} \lambda_c &= i_a[L_{ab0} + L_{aa2}\cos(2\theta - \frac{2\pi}{3})] + i_b[L_{ab0} + L_{aa2}\cos(2\theta - \pi)] \\ &- i_c[L_{aa0} + L_{aa2}\cos 2(\theta + \frac{2\pi}{3})] + i_{fd}L_{afd}\cos(\theta + \frac{2\pi}{3}) \\ &+ i_{kd}L_{akd}(\theta + \frac{2\pi}{3}) - i_{kq}L_{akq}\sin(\theta + \frac{2\pi}{3}) \end{aligned} \tag{2.4}$$

where

v_a, v_b, v_c: instantaneous stator phase to neutral voltages

i_a, i_b, i_c : instantaneous currents of the stator in phases *a, b, c*

v_{vf} : field voltage

$i_{fd}, i_{kd}, i_{kq1}, i_{kq2}$: field and amortisseur circuit currents

$R_a, R_{fd}, R_{kd}, R_{kq1}, R_{kq2}$: resistances of the stator and rotor circuits

$L_{aa}, L_{bb}.L_{cc}$: self-inductances of the stator winding

$L_{ab}, L_{bc}.L_{ca}$: mutual inductances between stator windings

$L_{afd}, L_{akd}.L_{akq}$: mutual inductances between stator and rotor windings

$L_{ffd}, L_{kkd}, L_{kk1q}, L_{kk2q}$: self-inductances of the rotor circuits

2.2 ROTOR EQUATIONS

The electrical model of rotor is presented in the equations (2.5) to (2.9).

$$\begin{bmatrix} v_{fd} \\ v_{1d} \\ v_{1q} \\ v_{2q} \end{bmatrix} = \begin{bmatrix} r_{fd} & 0 & 0 & 0 \\ 0 & r_{1d} & 0 & 0 \\ 0 & 0 & r_{1q} & 0 \\ 0 & 0 & 0 & r_{2q} \end{bmatrix} \begin{bmatrix} i_{fd} \\ i_{1d} \\ i_{1q} \\ i_{2q} \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} \lambda_{fd} \\ \lambda_{1dr} \\ \lambda_{1qr} \\ \lambda_{2qr} \end{bmatrix} \qquad (2.5)$$

where r_{fd}, r_{1d}, r_{1q}, r_{2q}, are the rotor resistances; $i_{fd}, i_{1d}, i_{1q}, i_{2q}$ are the rotor currents and $\lambda_{fd}, \lambda_{1dr}, \lambda_{1qr}, \lambda_{2qr}$ are the rotor flux linkage, defined by:

$$\lambda_{fd} = L_{ffd} i_{fd} + L_{fkd} i_{kd} - L_{afd}[i_a \cos\theta + i_b \cos(\theta - \frac{2\pi}{3}) + i_c \cos(\theta + \frac{2\pi}{3}) \qquad (2.6)$$

$$\lambda_{kd} = L_{fkd} i_{fd} + L_{kkd} i_{kd} - L_{akd}[i_a \cos\theta + i_b \cos(\theta - \frac{2\pi}{3}) + i_c \cos(\theta + \frac{2\pi}{3})] \quad (2.7)$$

$$\lambda_{k1q} = L_{kk1q} i_{k1q} + L_{akq}[i_a \cos\theta + i_b \cos(\theta - \frac{2\pi}{3}) + i_c \cos(\theta + \frac{2\pi}{3})] \quad (2.8)$$

$$\lambda_{k2q} = L_{kk2q} i_{k2q} + L_{akq}[i_a \cos\theta + i_b \cos(\theta - \frac{2\pi}{3}) + i_c \cos(\theta + \frac{2\pi}{3})] \quad (2.9)$$

2.3 MECHANICAL MODEL

The mechanical equations are given by:

$$\frac{d\theta}{dt} = \frac{2}{p}\omega \quad (2.10)$$

$$J\frac{2}{P}\frac{d\omega}{dt} = T_m - T_e - D\omega \quad (2.11)$$

$$T_e = \frac{3}{2}(\lambda_d i_q - \lambda_q i_d)\frac{p}{2} \quad (2.12)$$

where the number of magnetic poles per phase is p, $\omega = \frac{p}{2}\omega_{rm}$ is the rotor angular velocity expressed in electrical radians per second for a p-pole machine, ω_{rm} is the mechanical angular velocity of the rotor, J represents the inertia constant, T_m is the load torque , T_e is the electrical torque, and D is a mechanical damping coefficient.

2.4 DQ0 TRANSFORMATION

The previous equation are difficult to analyze, because it has different references for the stator and the rotor. The DQ0 transformation translates the stator equations into the rotor to the DQ reference in the rotor. The DQ reference is given by the saliency of the rotor poles as shown in Figure 6; the proposed reference is given in Figure 7. A representation of the synchronous generator referenced to the rotor frame is developed.

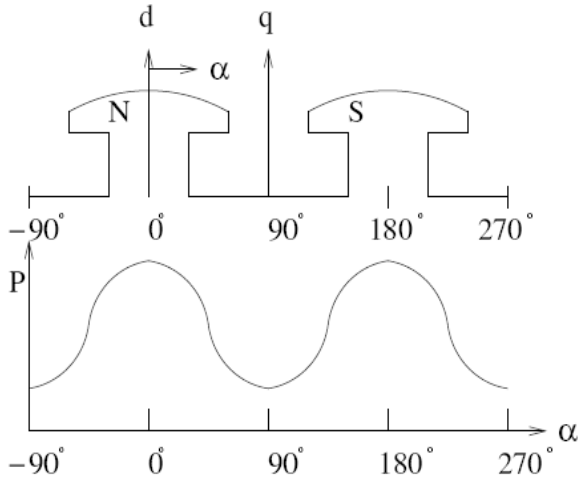

Figure 6 DQ mechanical references

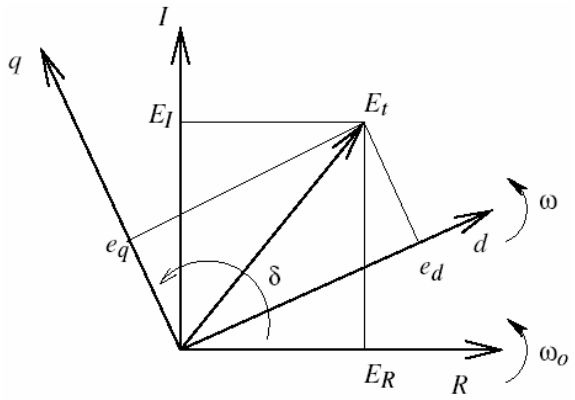

Figure 7 DQ0 reference

The general transformation from ABC to DQ0 reference is presented below.

$$\begin{bmatrix} v_0 \\ v_d \\ v_q \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \\ 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix} \begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} \quad (2.13)$$

Assuming magnetic linearity, the ABC quantities from the stator (2.1) gives the DQ0 equations presented below:

$$\begin{bmatrix} v_d \\ v_q \\ v_0 \end{bmatrix} = \begin{bmatrix} R_a & 0 & 0 \\ 0 & r_s & 0 \\ 0 & 0 & r_s \end{bmatrix} \begin{bmatrix} i_d \\ i_q \\ i_0 \end{bmatrix} - \omega \begin{bmatrix} \psi_{qs} \\ -\psi_{ds} \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} \psi_{ds} \\ \psi_{qs} \\ \psi_{0s} \end{bmatrix} \quad (2.14)$$

and for the rotor

$$\begin{bmatrix} v_{fd} \\ v_{1d} \\ v_{1q} \\ v_{2q} \end{bmatrix} = \begin{bmatrix} r_{fd} & 0 & 0 & 0 \\ 0 & r_{1d} & 0 & 0 \\ 0 & 0 & r_{1q} & 0 \\ 0 & 0 & 0 & r_{2q} \end{bmatrix} \begin{bmatrix} i_{fd} \\ i_{1d} \\ i_{1q} \\ i_{2q} \end{bmatrix} + \frac{d}{dt} \begin{bmatrix} \lambda_{fd} \\ \lambda_{1dr} \\ \lambda_{1qr} \\ \lambda_{2qr} \end{bmatrix} \quad (2.15)$$

where $r_s, r_{fd}, r_{1d}, r_{1q}, r_{2q}$ are the stator and rotor resistances; $i_d, i_q, i_{fd}, i_{1d}, i_{1q}, i_{2q}$, are the dq currents of the stator and rotor; $\lambda_{ds}, \lambda_{qs}, \lambda_{fd}, \lambda_{1dr}, \lambda_{1qr}, \lambda_{2qr}$ are the dq fluxes of the stator and rotor. For balanced systems it is usually assumed that the voltage and current i_o , v_o are equal to zero.

2.5 INCREMENTAL GENERATOR MODEL

As shown before (2.14) and (2.15) the equations require a large amount of data to be calculated (fluxes, currents, voltages, speed, and electrical parameters, in the DQ0 reference and all associated with transient and subtransient components) and also are necessary to include the mechanical parameters (2.10) to (2.12). The principal objective of this thesis is to calculate only the electrical parameters, for this reason it is necessary to make some simplifications as presented in [Jauregui, 2001] and [Burth, 1999].

The modeling assumptions are that the synchronous generator is connected to an infinite bus through a reactance x_e. The variations of the dq-axis fluxes $\frac{d\lambda_d}{dt}$ and $\frac{d\lambda_q}{dt}$ are usually neglected so the stator quantities will contain only fundamental frequency components. Another simplifying assumption normally made is that the per unit value of

the rotor velocity ω_r is equal to 1.0 in the stator voltages equations. This is not the same as saying that speed is constant; it assumes that speed changes are small and do not have a significant effect on the voltage.

Considering these assumptions, the incremental synchronous machine model for the electrical subsystem, linearized around an operating point is given by:

$$\frac{d\Delta E_q^{'}}{dt} = \frac{1}{T_{do}}\Delta E_q^{'} - \frac{xd - x'd}{T_{do}^{'} x_d^{''}}\Delta E_q^{''} + \frac{xd - x'd}{T_{do}^{'} x_d^{''}}\Delta V_q + \frac{k}{T_{do}^{'}}\Delta V_{fd} \qquad (2.16)$$

$$\frac{d\Delta Eq^{''}}{dt} = \left(\frac{1}{T_{do}^{''}} - \frac{1}{T_{do}^{'}}\right)\Delta E'q - \left(\frac{1}{T_{do}^{''}} + \frac{xd - x'd}{T_{do}^{'} x''d} + \frac{x'd - x''d}{T_{do}^{''} x''d}\right)\Delta E''q + \left(\frac{xd - x'd}{T_{do}^{'} x''d} + \frac{x'd - x''d}{T_{do}^{''} x''d}\right)\Delta Vq + \frac{k}{T_{do}^{'}}\Delta V_{fd} \qquad (2.17)$$

$$\frac{d\Delta E_d^{''}}{dt} = -\frac{xq}{T_{qo}^{''} x_q^{''}}\Delta E''d - \frac{xq - x''d}{T_{qo}^{''} x_q^{''}}\Delta V_d \qquad (2.18)$$

$$\Delta i_d = \frac{1}{x_d^{''}}(\Delta E_q^{''} - \Delta V_q) \qquad (2.19)$$

$$\Delta i_q = \frac{1}{x_q^{''}}\Delta E_d^{''} + \frac{1}{x_q^{''}}\Delta V_d \qquad (2.20)$$

where $\Delta V_d, \Delta V_q, \Delta V_{fd}$ are the incremental dq and field input voltages and $\Delta i_d, \Delta i_q$ are the incremental dq output currents. $\Delta E_q^{'}$ $\Delta E_q^{''}$ $\Delta E_d^{''}$, are the incremental dq transient and sub-transient voltages defined as $\Delta E_q^{'} \overset{\Delta}{=} \omega \frac{L_{md}}{L_{fd}}\lambda_{fd}$, $\Delta E_q^{''} \overset{\Delta}{=} \omega L_{md}^{''}\left(\frac{\lambda_{1qr}}{L_{1qr}} + \frac{\lambda_{2qr}}{L_{2qr}}\right)$ and

$\Delta E_q^{''} \overset{\Delta}{=} -\omega L_{mq}^{''}(\frac{\lambda_{1qr}}{L_{1qr}} + \frac{\lambda_{2qr}}{L_{2qr}})$. The parameters $x_d, x_q, x_d^{'}, x_q^{'}, x_d^{''}, x_q^{''}$ are the transient and sub-transient dq-axis reactances. $T_{do}^{'}, T_{do}^{''}$ are the d-axis transient and sub-transient time constants, k is the ratio of the mutual reactance of the direct axis and the field winding resistance given by $k = \frac{x_{md}}{r_{fd}}$.

The incremental mechanical equations are derived in [Kundur, 1994] and are given by

$$\frac{d\Delta\omega}{dt} = -\frac{D}{H}\Delta\omega - \frac{\omega_s}{2H}\Delta \mathrm{P} \qquad (2.21)$$

$$\frac{d\phi}{dt} = \Delta\omega \qquad (2.22)$$

$$\Delta\delta_t = \Delta\phi - \Delta\theta \qquad (2.23)$$

where ω_s is the synchronous machine frequency, and P is the output electric power.

From the equations (2.15) to (2.19), we can identify the parameters of the synchronous generator electrical subsystem, to be estimated.

$$\boldsymbol{\alpha} = [x_d \quad x_d^{'} \quad x_d^{''} \quad T_{d0}^{'} \quad T_{do}^{''} \quad k \quad x_q \quad x_q^{''} \quad T_{qo}^{''}]^T \qquad (2.24)$$

The electric subsystem described in (4.15) to (4.19), has nine parameters. It is common in the machine literature to have $x_d \quad x_d^{'} \quad x_d^{''} \quad T_{d0}^{'} \quad T_{do}^{''} \quad x_q \quad x_q^{''} \quad T_{qo}^{''}$ as the standard parameters [Kundur, 1994]. There is an additional parameter k in our model, which relates the field winding with the direct axis.

CHAPTER 3 BATCH PARAMETER ESTIMATION

The parameter identification problem is the process to estimate the parameters of a given model structure. Figure 8 represents the general structure used for output error parameter estimation, where the error between the estimated output and the measured output is used to adjust the model parameters. The parameter obtained by this estimation procedure select the parameters estimated by minimizing the corresponding cost function. The resulting optimization problems usually have the following characteristics [Walter et al, 1994]:

- The number of parameters to be optimized is small, typically less than ten.
- The cost function is smooth, the fist and second derivatives are relatively easy to compute.
- Optimization is unconstrained (although the parameters might belong to some simple-shaped prior feasible set such as a box).
- The effects of the various parameters on the value of the cost are very unequal. The problem is ill conditioned.
- The problem is not convex and local optimizer may exist that do not correspond to the best possible value of the cost; however there are some suitable experiments that can eliminate such parasitic local optimizer.

The identification problem presented in [Walter et al, 1994] requires the definition of the cost function. The cost function usually is selected as a Least Square or Mean

Squares. The batch and recursive identification methodologies presented here use the least square cost function.

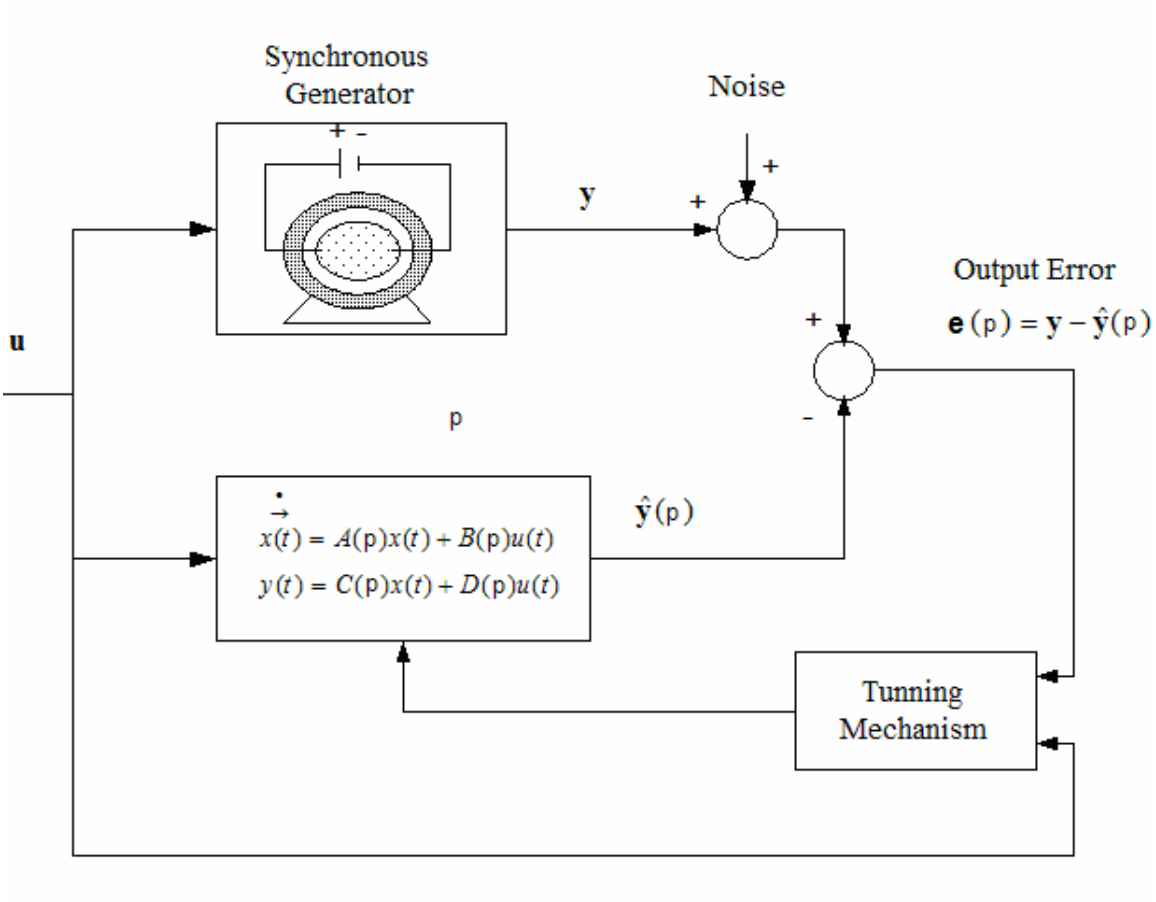

Figure 8 General parameter identification structures.

3.1 INTRODUCTION

Many studies have been published showing different approaches for batch parameter estimation in synchronous generators; [Jin-Cheng et al, 1994], [Robet el al, 1995] and [Jauregui, 2001]. These methodologies present a batch estimation using Least Squares and its modifications suggested increasing the convergence or reducing the ill conditioning. In this section, a brief introduction of batch estimation is presented: the

Gauss-Newton and Leverberg-Marquardt algorithm, and simulations showing the performance of these algorithms.

3.2 LINEAR LEAST SQUARES

First is necessary to define a square cost for arbitrary error function of the parameters (p).

$$j(p) = e^T(p)Qe(p) \tag{3.1}$$

where Q is a positive definite weighting matrix. As presented in Figure 8, if the error to be minimized corresponds to the difference between the estimated and the measured output.

$$e(p) = y - y(p) \tag{3.2}$$

and the system to be minimized is in the form

$$y(p) = Rp \tag{3.3}$$

The values of p which minimizes 3.1 called "p estimated ($\hat{p}$)" is obtained by differentiating it with respect to p and equaling to zero and solving for p.

$$\frac{\partial j}{\partial p} = -2R^T Q(y - Rp) = 0 \tag{3.4}$$

and solved for p

$$\hat{p} = (R^T QR)^{-1} R^T Qy \tag{3.5}$$

However if the matrix R^TQR is ill conditioned this solution will be very sensitive to noise in the measurements.

3.3 GAUSS-NEWTON METHOD.

The Gauss-Newton method used in the model in (3.2) is nonlinear. The expansion of the cost (3.1) about the estimated at the k[th] iteration $\hat{p}^k$ is:

$$j(\hat{p}^{k+1}) = j(\hat{p}^k + \Delta p) = j(\hat{p}^k) + g^T(\hat{p}^k)\Delta p + \frac{1}{2}\Delta p^T H(\hat{p}^k)\Delta p + o(\|\Delta p\|^2) \quad (3.6)$$

where $g(\hat{p}^k)$ represents the gradient of the cost

$$g(\hat{p}^k) = \frac{\partial j}{\partial p}\Big|_{p=\hat{p}k} = -J^{(k)T}(y-\hat{y}) \qquad with \quad J^{(k)} = \frac{\partial \hat{y}}{\partial p}\Big|_{p=\hat{p}k} \quad (3.7)$$

where J is the Jacobian matrix and H is the Hessian matrix. If the Hessian is approximated only by the first order elements, then

$$H_a(p) = \frac{\partial^2 j}{\partial p \partial p^T}\Big|_{p=\hat{p}} \approx \sum_{i=1}^{nt} w_i \frac{\partial e(t_i,p)}{\partial p}\frac{\partial e(t_i,p)}{\partial p^T} = J^T J \quad (3.8)$$

Defining the steep size as $\Delta j = j(\hat{p}^{k+1}) - j(\hat{p}^k)$ and calculating the value of Δp that decreases the cost and satisfies the optimal condition:

$$\frac{\partial \Delta j}{\partial \Delta p}\Big|_{\Delta\hat{p}} = 0 \approx H(\hat{p}^k)\Delta\hat{p}^k + g(\hat{p}^k) \quad (3.9)$$

From (3.9), and assuming that the Hessian is invertible it is possible to calculate the step Δp as:

$$\Delta\hat{p}^k = H^{-1}(\hat{p}^k)g(\hat{p}^k) \quad (3.10)$$

and adding a step-length λ_k, the general equation that defines the Gauss Newton algorithm is:

$$\hat{p}^{k+1} = \hat{p}^k - \lambda_k (J^T J)^{-1} J^T (y-\hat{y}) \quad (3.11)$$

The Gauss Newton algorithm presented above was applied to the model described in Section 2. The per unit parameters of synchronous generator of 100 MVA were taken from [Zhao,1995] which are presented in Table 1.

Table 1 Nominal parameters of synchronous generator

Parameters	x_d	$x_d^{'}$	$x_d^{''}$	$T_{do}^{'}$	$T_{do}^{''}$	k	x_q	$x_q^{''}$	$T_{qo}^{''}$
Nominal Values	1.414	0.333	0.208	5.85	0.194	1552	1.302	0.396	0.955

The input signal to the system is presented in Figure 9. and the output in Figure 10.

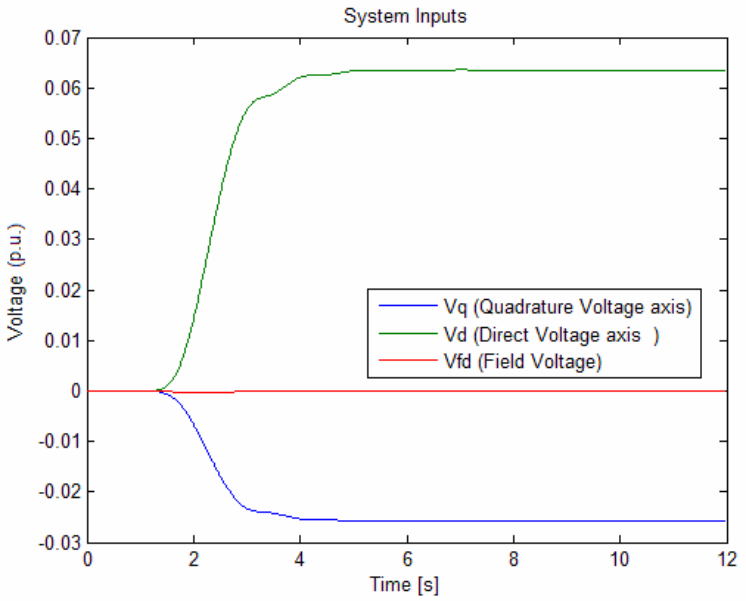

Figure 9. Input to the Gauss 100 MVA synchronous generator

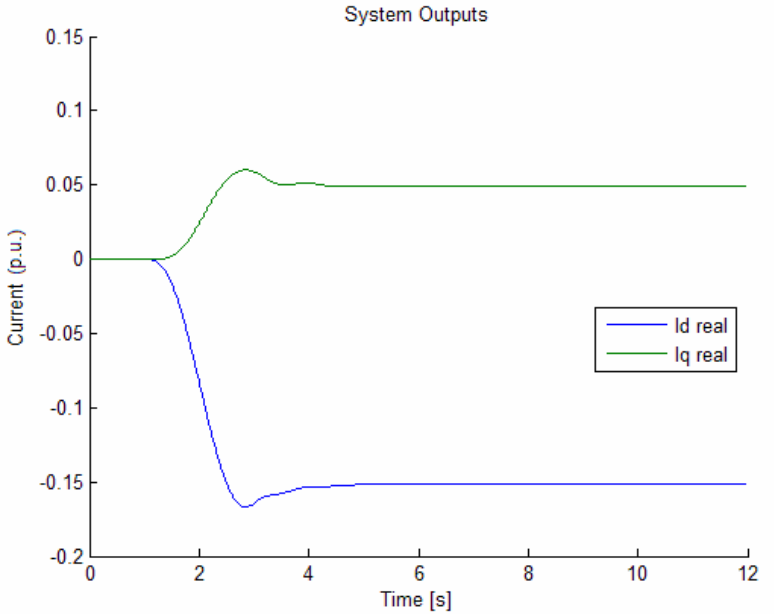

Figure 10. Output to the 100 MVA synchronous generator

The Gauss-Newton algorithm was applied to the noise free case and also for the noisy case. For the noisy simulation the wave forms used are presented in Figure 11 and Figure 12. The noisy has a maximum value of: 0.005 in pu for Vd and Vq, 0.0005 for Vfd, and 0.001 for Id and Iq.

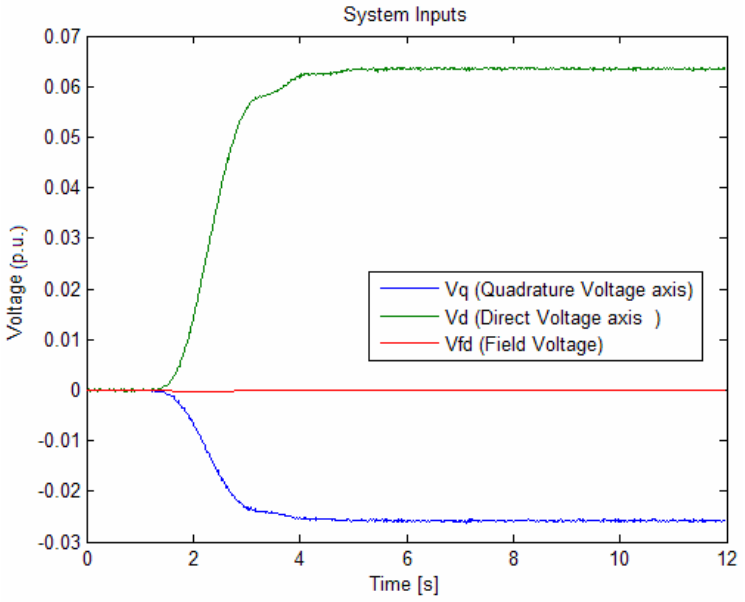

Figure 11. Input to the 100 MVA synchronous generator (noisy case)

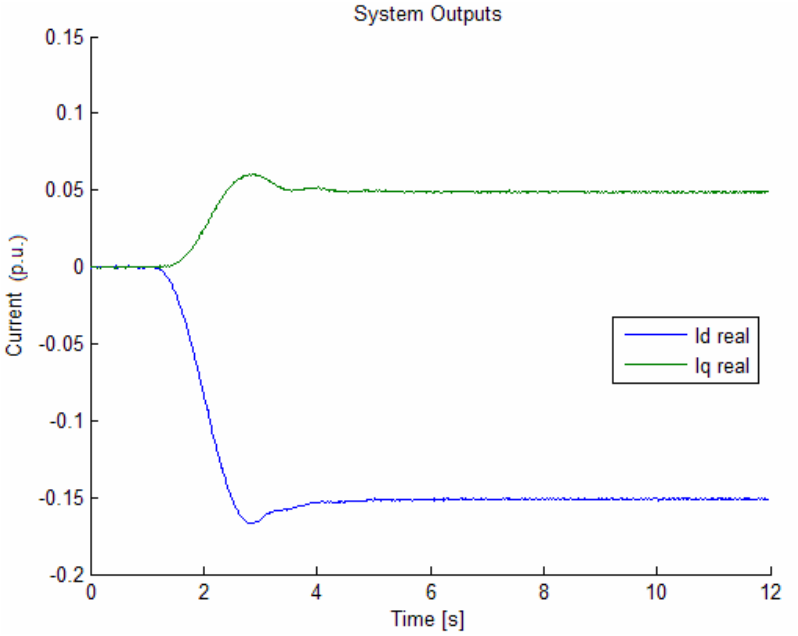

Figure 12. Output to the 100 MVA synchronous generator (noisy case)

The result of the application of Gauss Newton parameter estimation for the noise free case is presented in Figure 13. The method converges for an initial estimated of 90% of the true values, and also is possible to give a worst initial value, however to compare with the recursive estimator this value is selected. The method converge for all parameters, however is important to observe it is not monotonically for the parameters K and T_{d0}' as for the other parameters.

The results with noise are presented in Figure 14. In this simulation, the ill conditioning problem causes nonconvergence of the method. For this case 6 of 9 parameters tend to infinity and the remaining converge and only two are closer to the real value.

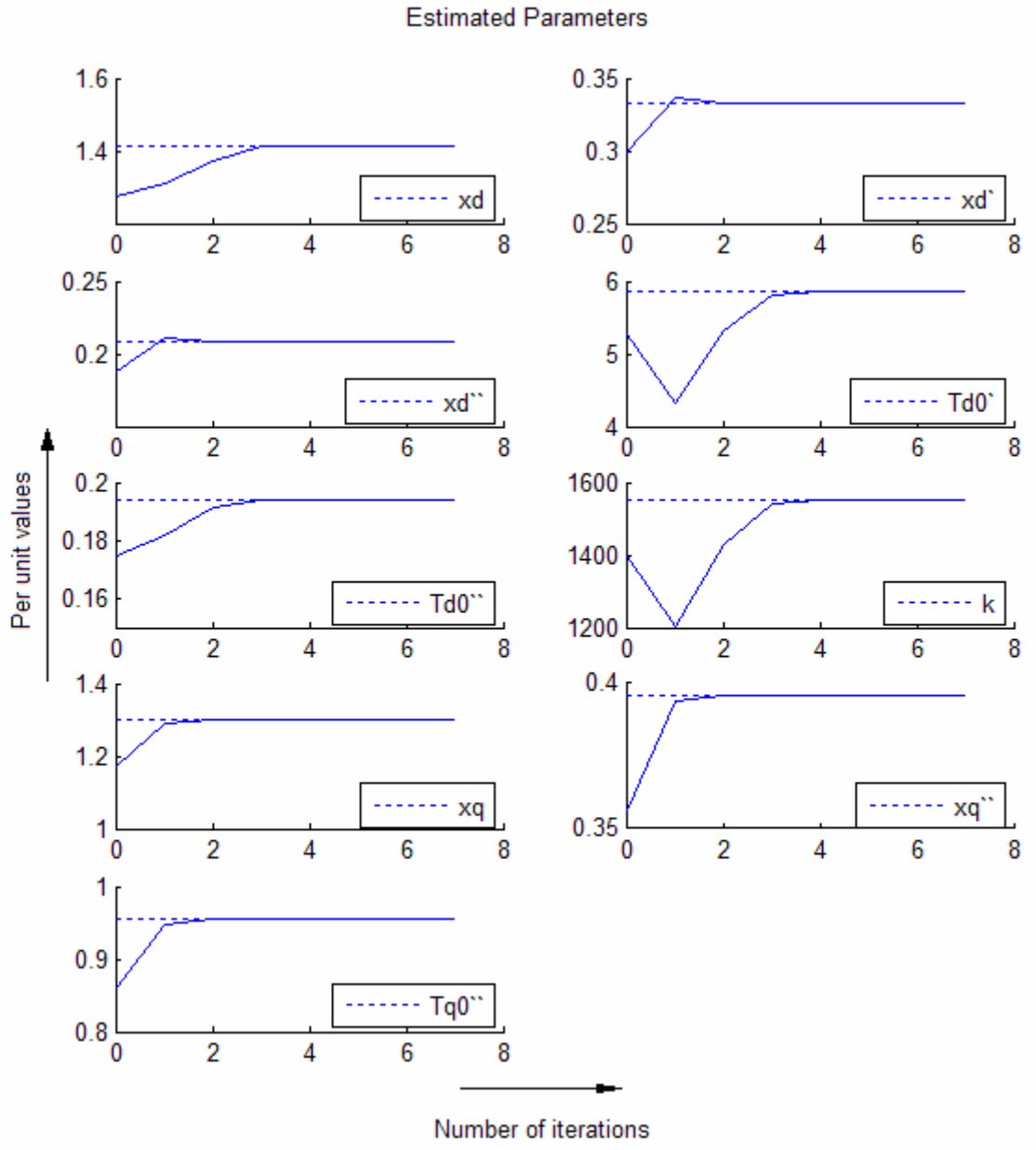

Figure 13 Estimated Parameters applying Gauss-Newton (noise free case)

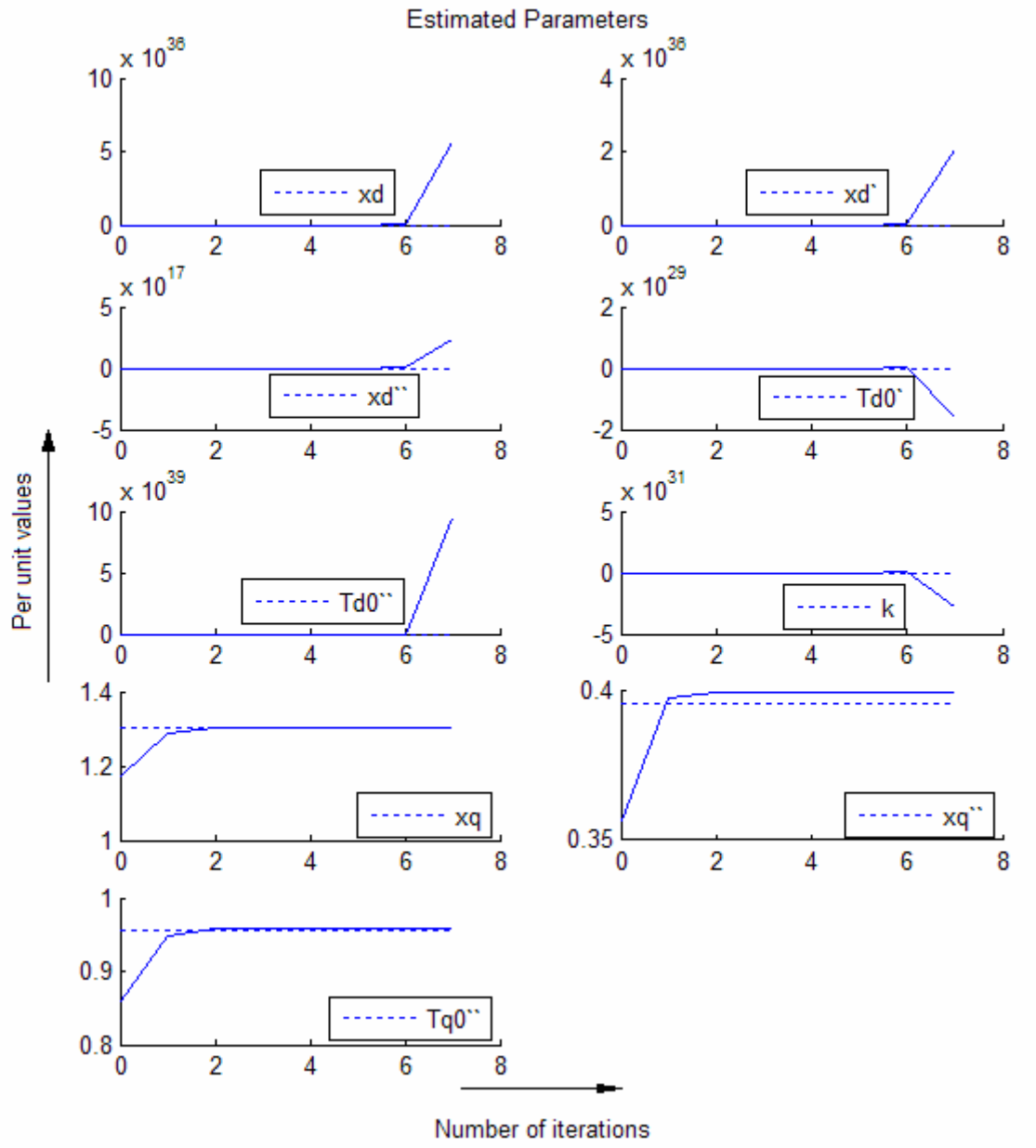

Figure 14 Estimated Parameters applying Gauss-Newton (noisy case)

3.4 LEVENBERG-MARQUARDT METHOD

The Levenberg-Marquardt method is a modification to the Gauss Newton method where Δp is calculated from using

$$[H\ (\hat{p}^k) + \mu_k I]\Delta p = -g(\hat{p}^k) \qquad (3.12)$$

Instead of (3.9), when μ_k=0 the method is equal to the Gauss-Newton and if μ_k tends to infinity it tends to the gradient method. The advantage of his method is the possibility adjusting μ_k at every iteration.

The suggested criterion to adapt μ_k is presented below:

- If $j(\hat{p}^{k+1}) \leq j(\hat{p}^k)$, then μ_k must divided by 10 (the method converges, and approximate to the Gauss Newton method).

- Else μ_k must multiplied by 10 and $\hat{p}^{k+1}$ is rejected (the method diverges, and approximate to the gradient method).

The regularization of Leveberg-Marquardt was applied for the parameter identification problem. The noise free Figure 15 and the noisy Figure 16 cases were analyzed.

Figure 15 present a slow convergence of Levenberg-Marquardt (LM) method compared with the Gauss Newton (GN). The required iterations are 15 to reach the nominal value, compared with 7 for the GN. The noisy analysis present the LM estimator which don't converge to the nominal values, however compared with the GN the LM is more stable because the estimated parameters don't go to infinity and remain constant during some iterations. In these results the decoupling between the direct axis parameters and quadrature axis is shown. The direct axis parameters do not convergence and the quadrature approximate to the nominal value. These results shows the regularization as a good alternative to stabilized the identification algorithm and suggest the analysis of a decoupled method that work in two separated blocks (direct and quadrature axis parameters).

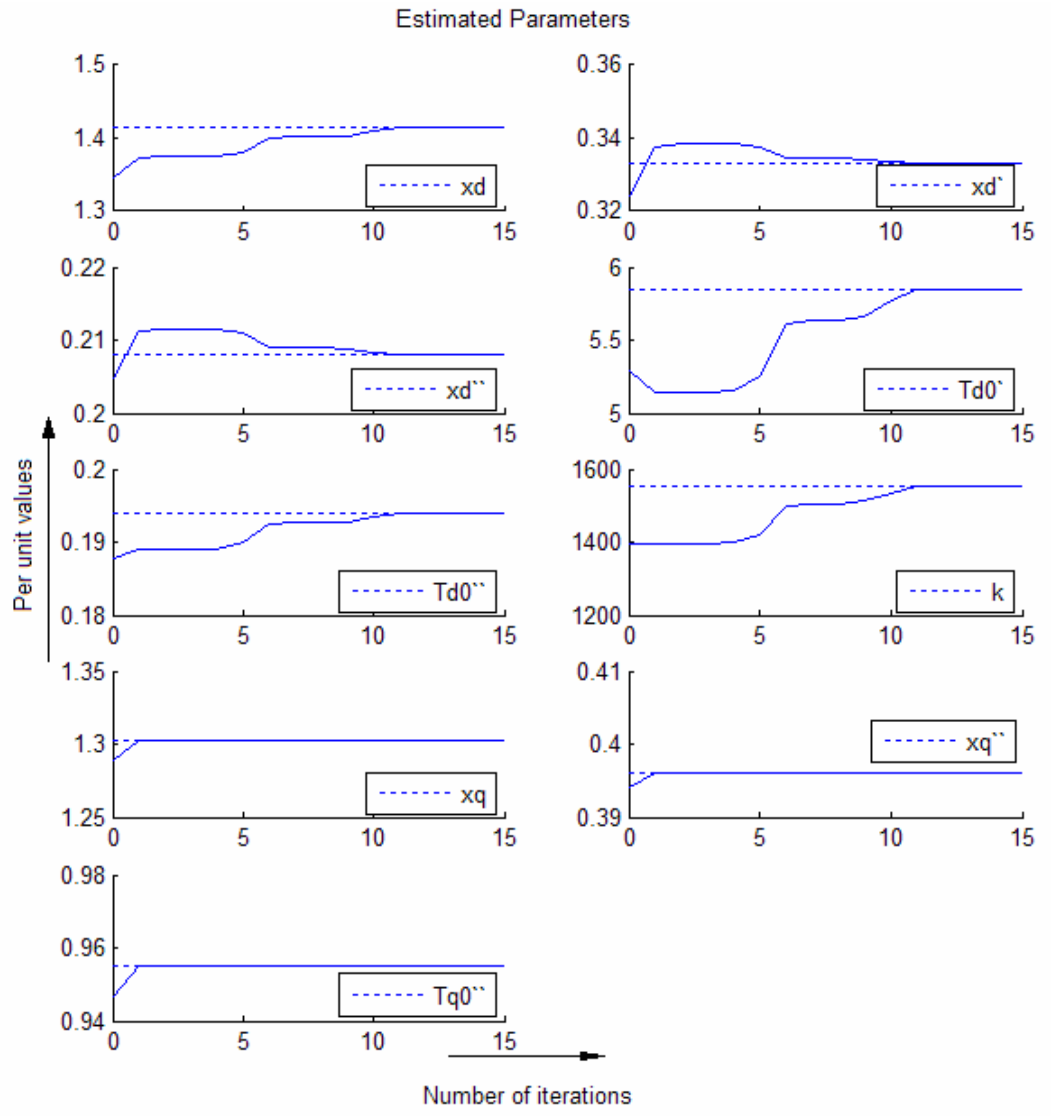

Figure 15 Estimated Parameters applying Levenberg-Marquardt (noise free case)

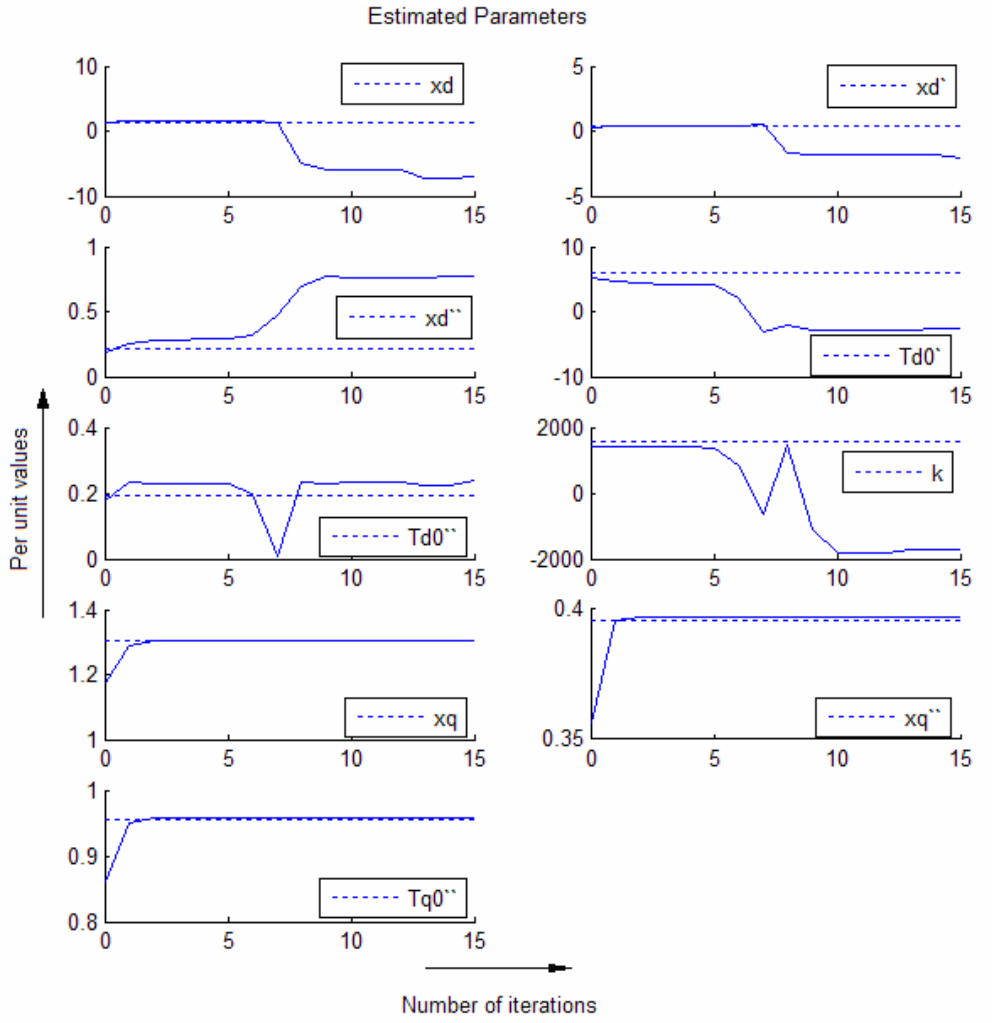

Figure 16 Estimated Parameters applying Levenberg-Marquardt (noisy case)

CHAPTER 4 EXTENDED KALMAN FILTER

The recursive parameter identification techniques reduce memory space required to store information of events, because it requires present measurements of electrical variables, the past measurements are represented by the covariance matrix (smaller dimension than the measurements). This technique allows detection in real time of changes in electrical parameters by the identification process.

The Extended Kalman Filter (EKF) algorithm is used for recursive parameter identification. The principal advantage of EKF is the simultaneous estimation of parameters and states.

4.1 ALGORITHM DESCRIPTION

Under the appropriate model assumptions the Extended Kalman Filter presented in [Gelb,1974] is a minimum variance estimator. The general procedure to get the EKF is presented below. First define the linear system:

$$\begin{aligned} \dot{x}(t) &= f(x(t),t) + w(t) \\ z(t) &= h(x(t)) + v(t) \end{aligned} \tag{4.1}$$

The previous system is described for continuous time. If the process is sampled at time t_{k-1} and no measurements are taken until t_k. The states and the output can be propagated as:

$$x(t) = x(t_{k-1}) + \int_{t_{k-1}}^{t} f(x(\tau),\tau)d\tau + \int_{t_{k-1}}^{t} w(\tau)d\tau \tag{4.2}$$

$$z_k(t) = h_k(x(t)) + v_k(t) \tag{4.3}$$

Taking the expectation of equation (4.2) and differentiating produces

$$\frac{dE[x(t)]}{dt} = E[f(x(t),t)] \qquad t_{k-1} \le t < t_k \tag{4.4}$$

with the condition

$$E[x(t_{k-1})] = \hat{x}(t_{k-1}) \tag{4.5}$$

in the interval $t_{k-1} \le t < t_k$ the mean of x(t) is the solution to (4.3), which can be written:

$$\dot{\hat{x}}(t) = \hat{f}(x(t),t) \qquad t_{k-1} \le t < t_k \tag{4.6}$$

The above result can be used to calculate the error covariance matrix:

$$P(t) \equiv E\left[[\hat{x}(t) - x(t)][\hat{x}(t) - x(t)]^T\right] \tag{4.7}$$

substituting (4.2) in (4.7) and differentiating the result is:

$$\dot{P}(t) = \hat{xf}^T - \hat{x}\hat{f}^T + \hat{fx}^T - \hat{f}\hat{x}^T + Q(t) \qquad t_{k-1} \le t < t_k \tag{4.8}$$

The equations (4.6) and (4.8) are the propagation equations of the covariance and the states for the linear estimation problem. To show the dependence of $\dot{\hat{x}}(t)$ on F(t) and x(t) the linear case of the system (4.6) is analysed.

$$\dot{\hat{x}}(t) = F(t)\hat{x}(t) = F(t)\hat{x}(t) = f(\hat{x}(t),t) \tag{4.9}$$

The same result can be obtained from (4.8) which only depends on F(t), Q(t) and P(t).). However the analyzed system is not linear; in order to calculate the error covariance matrix and the states by integration of (4.6) and (4.8) it is necessary to make some simplifications and assumptions. The function f(x(t),t) expanded in Taylor series is:

$$f(x,t) = f(\hat{x},t) + \left.\frac{\partial f}{\partial x}\right|_{x=\hat{x}} (x - \hat{x}) + \ldots \tag{4.10}$$

taking the expectation of equation (4.10)

$$\hat{f}(x,t) = f(\hat{x},t) + 0 + ... \qquad (4.11)$$

taking a first order approximation and substituting 4.11 in 4.6 produces

$$\dot{\hat{x}}(t) = f(\hat{x}(t),t) \qquad t_{k-1} \le t < t_k \qquad (4.12)$$

The same approximation is applied to the differential equation for the estimation of error covariance matrix (4.8)

$$\dot{P}(t) = F(\hat{x}(t),t)P(t) + P(t)F^T(\hat{x}(t),t) + Q(t) \qquad t_{k-1} \le t < t_k \qquad (4.13)$$

the function $F(\hat{x}(t),t)$ is the Jacobian whose ij[th] elements is given by

$$f_{ij}(\hat{x}(t),t) \equiv \left.\frac{\partial f_i(x(t),t)}{\partial x_j}\right|_{x=\hat{x}} \qquad (4.14)$$

the equations (4.12) and (4.13) are approximate expressions used to propagate the states and the covariance matrix. In order to develop an algorithm it is necessary to make a representation of the system in discrete time and also include the measurements in the updating equation. Consider the updating equation of a linear discrete system:

$$\hat{x}_k(+) = a_k + K_k z_k \qquad (4.15)$$

The values "a" and "K" must be calculated. The errors of the estimated value after and before the time of sampling are defined by:

$$\begin{aligned} \tilde{x}_k(+) &\equiv \hat{x}_k(+) - x_k \\ \tilde{x}_k(-) &\equiv \hat{x}_k(-) - x_k \end{aligned} \qquad (4.16)$$

substituting (4.15) and (4.3) in (4.16) is possible to calculate the error of the estimated state after the sample time t_k

$$\tilde{x}_k(+) \equiv a_k + K_k h_k(x_k) + K_k v_k + \tilde{x}_k(-) - \hat{x}_k(-) \qquad (4.17)$$

using the requirements of an unbiased estimator and calculating the estimated value of equation (4.17)

$$0 = a_k + K_k \hat{h}_k(x_k) - \hat{x}_k(-) \tag{4.18}$$

eliminating a_k in (4.15) by substituting (4.18) gives:

$$\hat{x}_k(+) = \hat{x}_k(-) + K_k\left[z_k - \hat{h}_k(x_k)\right] \tag{4.19}$$

to express the estimation error it is necessary to combine equations (4.17) and (4.18)

$$\tilde{x}_k(+) = \tilde{x}_k(-) + K_k\left[h_k(x_k) - \hat{h}_k(x_k)\right] + K_k v_k \tag{4.20}$$

to calculate the estimated value of the state it is necessary to determine the optimal Kalman Gain K_k. The Kalman Gain is selected to minimize the error covariance matrix. The covariance matrix is:

$$P_k(+) = E\left[\tilde{x}_k(+)\tilde{x}_k(+)^T\right] \qquad and \quad P_k(-) = E\left[\tilde{x}_k(-)\tilde{x}_k(-)^T\right] \tag{4.21}$$

Assuming that P_k is independent of v(t), and R_k=E[$v_k v_k^T$] the covariance matrix is:

$$\begin{aligned} P_k(+) = P_k(-) &+ K_k E\left[\left[h_k(x_k) - \hat{h}_k(x_k)\right]\left[h_k(x_k) - \hat{h}_k(x_k)\right]^T\right] K_k^T \\ &+ E\left[\tilde{x}_k(-)\left[h_k(x_k) - \hat{h}_k(x_k)\right]^T\right] K_k^T \\ &+ K_k E\left[\left[h_k(x_k) - \hat{h}_k(x_k)\right]\tilde{x}_k(-)^T\right] + K_k R_k K_k^T \end{aligned} \tag{4.22}$$

defining the trace of the error of covariance matrix as

$$trace(P_k(+)) = E\left[\tilde{x}_k(+)^T \tilde{x}_k(+)\right] \qquad and \qquad \frac{\partial trace(P_k(+))}{\partial K_k} = 0 \tag{4.23}$$

differentiating and calculating the trace of (4.22), including (4.23) and solving for K_K

$$K_k = -E\left[\tilde{x}_k(-)\left[h_k(x_k) - \hat{h}_k(x_k)\right]^T\right] * \left\{E\left[\left[h_k(x_k) - \hat{h}_k(x_k)\right]\left[h_k(x_k) - \hat{h}_k(x_k)\right]^T\right] + R_k\right\}^{-1} \quad (4.24)$$

substituting (4.24) in (4.22)

$$P_k(+) = P_k(-) + K_k E\left[\left[h_k(x_k) - \hat{h}_k(x_k)\right]\tilde{x}_k(-)^T\right] \quad (4.25)$$

The equations (4.19), (4.24) and (4.25) give the discrete time algorithm. However the function $\hat{h}_k$ must be calculated. The function h_k is expanded in Taylor series as:

$$h_k(x_k) = h_k(\hat{x}_k) + H_k(\hat{x}_k(-))(x_k - \hat{x}_k(-)) \ \ldots \quad (4.26)$$

where

$$H_k(\hat{x}_k(-)) = \frac{\partial h_k(x_k)}{\partial x} \quad (4.27)$$

Is the Jacobian of h with respect to x. The general algorithm can be summarized in equations (4.22-30) which are obtained by truncating (4.26) only to two first terms and substituting in (4.24) and (4.25)

$$\hat{x}_k(+) = \hat{x}_k(-) + K_k\left[z_k - \hat{h}_k(x_k)\right] \quad (4.28)$$

$$K_k = P_k(-)H_k^T(\hat{x}_k(-))\left[H_k^T(\hat{x}_k(-))P_k(-)H_k(\hat{x}_k(-)) + R_k\right]^{-1} \quad (4.29)$$

$$P_k(+) = \left[I - K_k H_k(\hat{x}_k(-))\right]P_k(-) \quad (4.30)$$

The previous equations don't give information about the dependence of the error covariance matrix and the states from the parameters $\hat{p}_k$ and do not include the input u_k. In order to generate a complete algorithm as presented in [Walter et al, 1994] is necessary to represent the system in the form:

$$x_{k+1} = f(x_k, u_k, p_k) + v_k \quad (4.31)$$

$$z_k = h(x_k, p_k) + w_k \tag{4.32}$$

where u_k and v_k and w_k are white noise, with the initial condition:

$$x_0 = x_0(p_0) \tag{4.33}$$

A common approach to apply the EKF to parameter estimation is to augment the system state with the parameters to be estimated as follows:

$$x_k^e = \begin{bmatrix} x_k \\ p_k \end{bmatrix} \tag{4.34}$$

using the equation (4.31), is possible to rewrite the state evolution equation of the extended state as:

$$x_{k+1}^e = \begin{bmatrix} f(x_k, u_k, p_k) \\ p_k \end{bmatrix} + \begin{bmatrix} v_k \\ v_k^p \end{bmatrix} = f^e(x_k^e, u_k, p_k) + v_k^e \tag{4.35}$$

expanding $f^e(x_k^e, u_k, p_k)$ by Taylor series and taking the first order elements:

$$f^e(x_k^e, u_k) = f^e(\hat{x}_{k|k}^e, u_k) + A_k(\hat{x}_k^e - \hat{x}_{k|k}^e) \tag{4.36}$$

with

$$A_k = \frac{\partial f^e(x_k^e, u_k)}{\partial x_k^{eT} \mid \hat{x}_{k|k}^e} = \begin{bmatrix} \dfrac{\partial f(x_k, u_k, p_k)}{\partial x_k^T} & \dfrac{\partial f(x_k, u_k, p_k)}{\partial p_k^T} \\ 0 & I \end{bmatrix} \Big| \hat{x}_{k|k}^e \tag{4.37}$$

and replacing $h^e(x_k^e)$ by its first order expansion around $\hat{x}_{k|k-1}^e$

$$h^e(x_k^e) = h^e(\hat{x}_{k|k-1}^e) + C_k(x_k^e - \hat{x}_{k|k-1}^e) \tag{4.38}$$

with

$$C_k = \frac{\partial h^e(x_k^e)}{\partial x_k^{eT} \mid \hat{x}_{k|k-1}^e} = \begin{bmatrix} \dfrac{\partial h(x_k, p_k)}{\partial x_k^T} & \dfrac{\partial h(x_k, p_k)}{\partial p_k^T} \end{bmatrix} \Big| \hat{x}_{k|k-1}^e \tag{4.39}$$

the non-linear state and observation equations are then approximated by:

$$x_{k+1}^e = A_k x_k^e + f(\hat{x}_{k|k}^e, u_k) - A_k \hat{x}_{k|k}^e + v_k^e \quad (4.40)$$

$$z_k = C_k x_k^e + h(\hat{x}_{k|k-1}^e) - C_k \hat{x}_{k|k-1}^e + w_k \quad (4.41)$$

Combining the equations (4.40) and (4.41) and using the general form of the propagation equations for the state and the error covariance matrix (4.28), (4.29) and (4.30) it is possible to define the general algorithm for the Extended Kalman Filter:

$$\hat{x}_{k+1|k} = f(\hat{x}_{k|k}, u_k) \quad (4.42)$$

$$P_{k+1|k} = A_k P_{k|k} A_k^T + V^e \quad (4.43)$$

$$K_{k+1} = P_{k+1|k} C_{k+1}^T (W + C_{k+1} P_{k+1|k} C_k^T)^{-1} \quad (4.44)$$

$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1} C_{k+1} P_{k+1|k} \quad (4.45)$$

and the state estimate updated by:

$$\hat{x}_{k+1|k+1}^e = \hat{x}_{k+1|k}^e + K_{k+1}[z_{k+1} - h^e(\hat{x}_{k+1|k}^e)] \quad (4.46)$$

The previous equations have the parameter "v_k^e" "w_k^e", which commonly are selected as diagonal matrices. The bigger value in the diagonal of "v_k^e" produces a faster change in the predicted value of x, and the bigger value in the diagonal of "w_k^e" produces that the changes in the output "Z" are taken less in account for the estimator. The flowchart presented in Figure 17 summarizes the extended Kalman Filter algorithm.

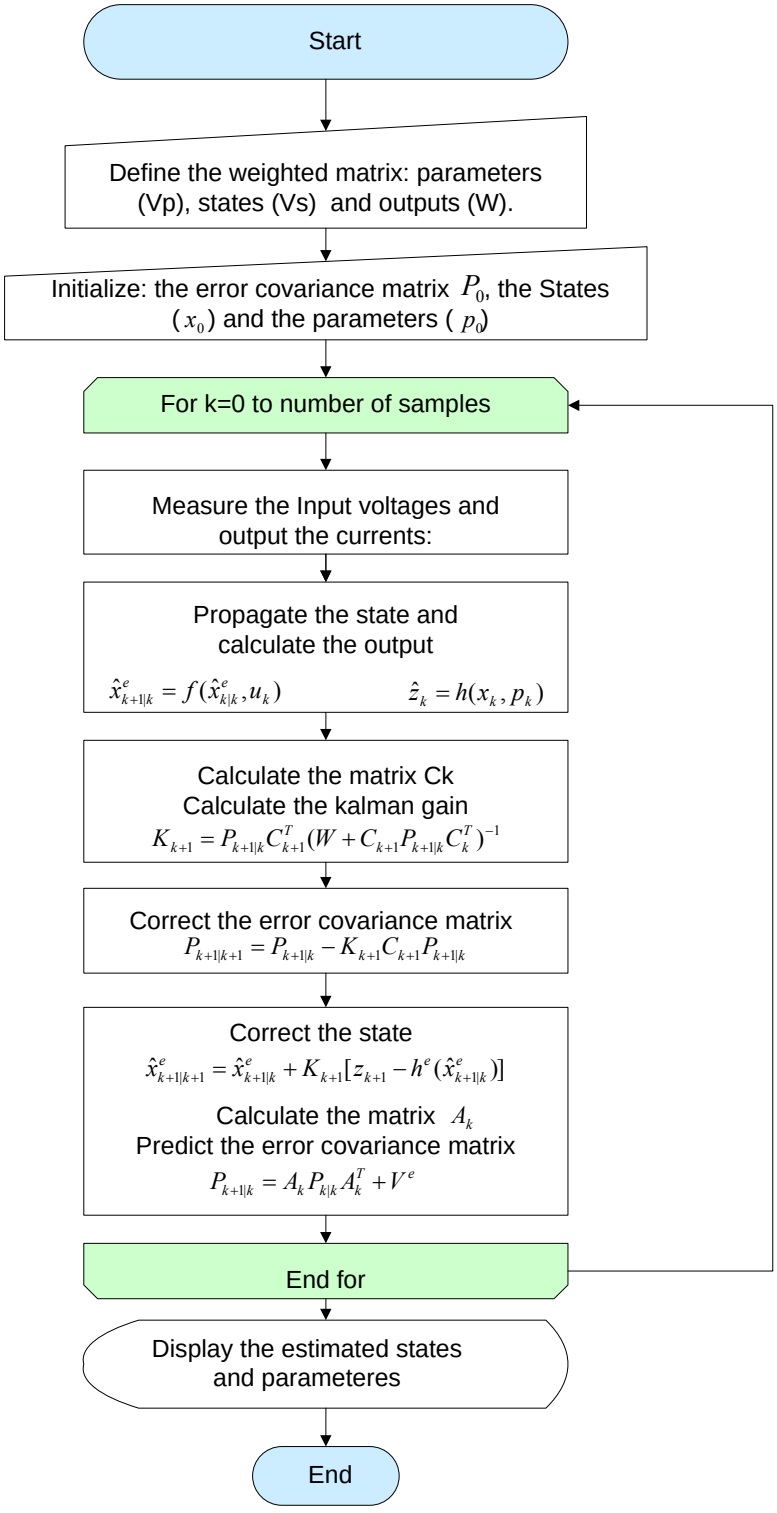

Figure 17. Extended Kalman Filter Flowchart

4.2 EXTENDED KALMAN FILTER APPLIED TO SYNCHRONOUS GENERATOR MODEL

The Extended Kalman Filter was applied to state and parameter estimation of the Synchronous Generator model presented in Section 3. The subfunction "obs_kal.m" presented in the Appendixes A and C compute the matrixes A_k and C_k which are required by the EKF. The function "esta0.m" propagates the state and calculates the error between the estimated output and the measured output. The program "rec_kal_ext.m" implements the general EKF algorithm.

Two studies were performed for the EKF. The first study looks at the convergence of EKF when the measured signals are noise free, and the second with measured signals with a 5% Gaussian noise. The states, outputs, and parameters from the noise free analysis are presented in Figure 18 to Figure 20, respectively. The noisy analysis was simulated with the input and output signals presented in Figure 11 and Figure 12. The estimated parameters, states and outputs for the noisy case are presented in Figure 21, to Figure 23 . Table 2 summarizes the results of the two first analysis of the EKF.

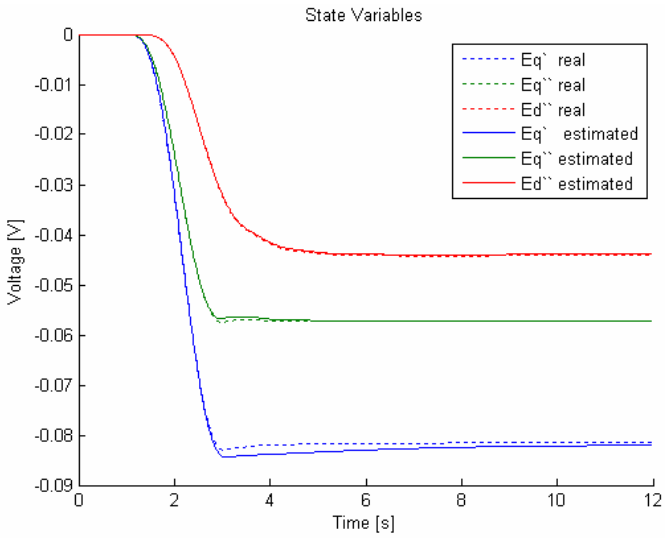

Figure 18. Estimated States EKF full order (noise free case)

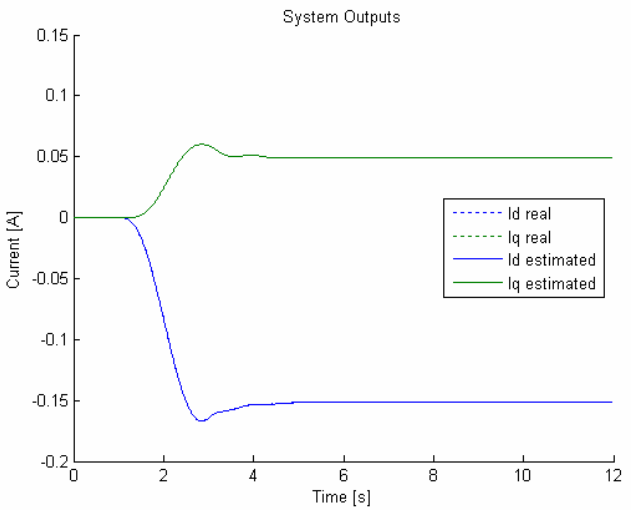

Figure 19. Estimated output EKF full order (noise free case)

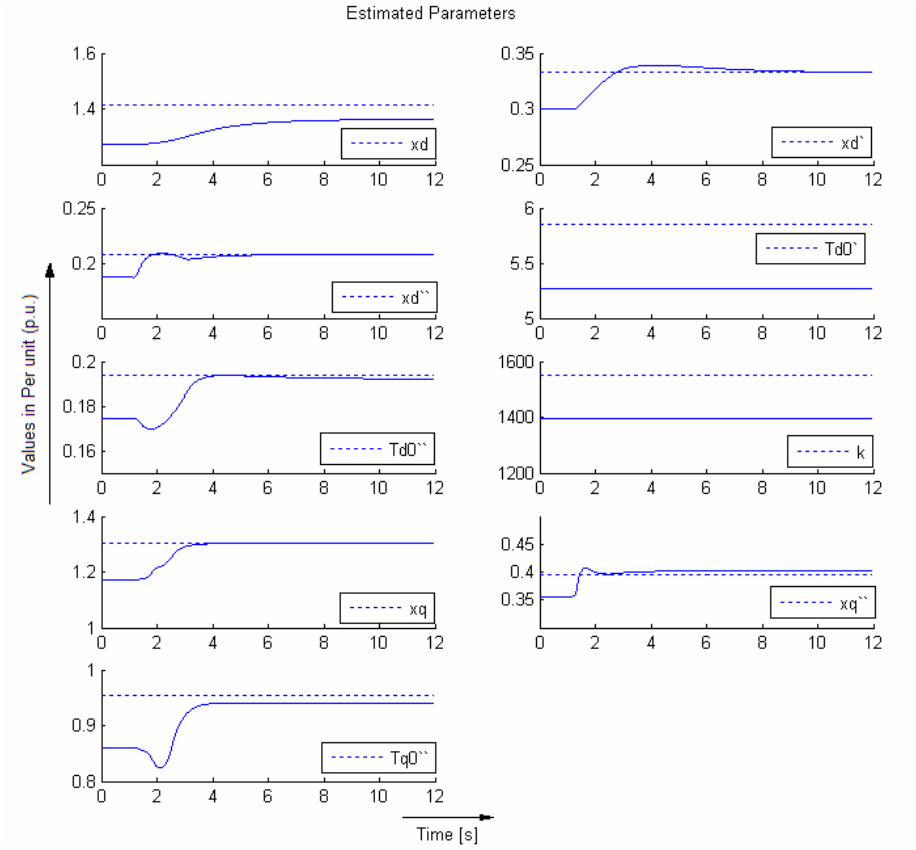

Figure 20. Estimated parameters EKF full order (noise free case)

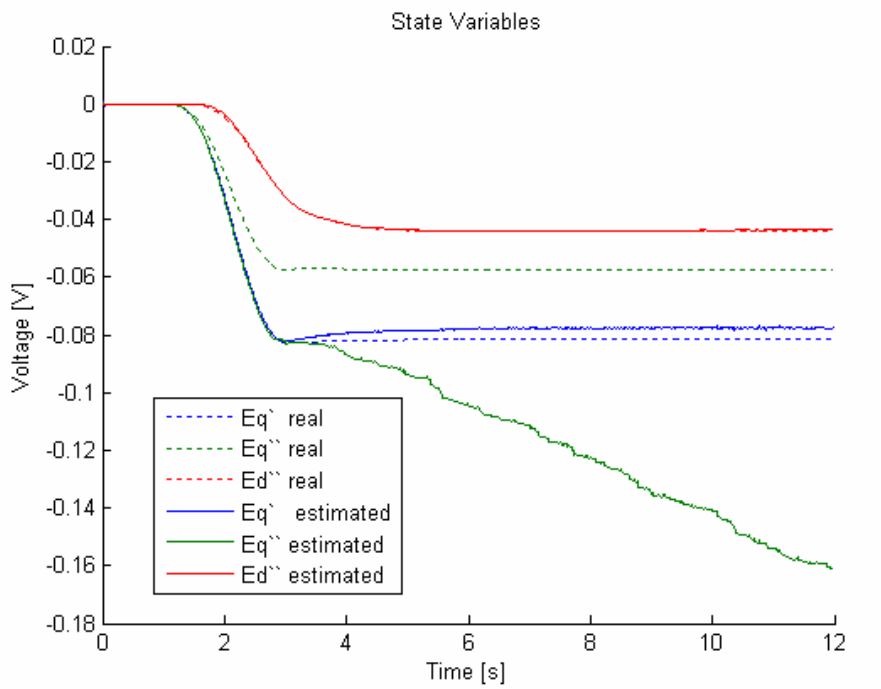

Figure 21. Estimated States EKF full order (noisy case)

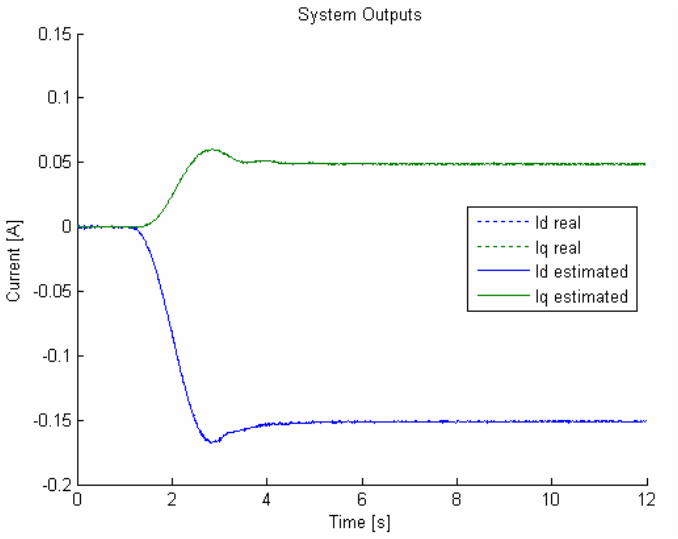

Figure 22. Estimated output EKF full order (noisy case)

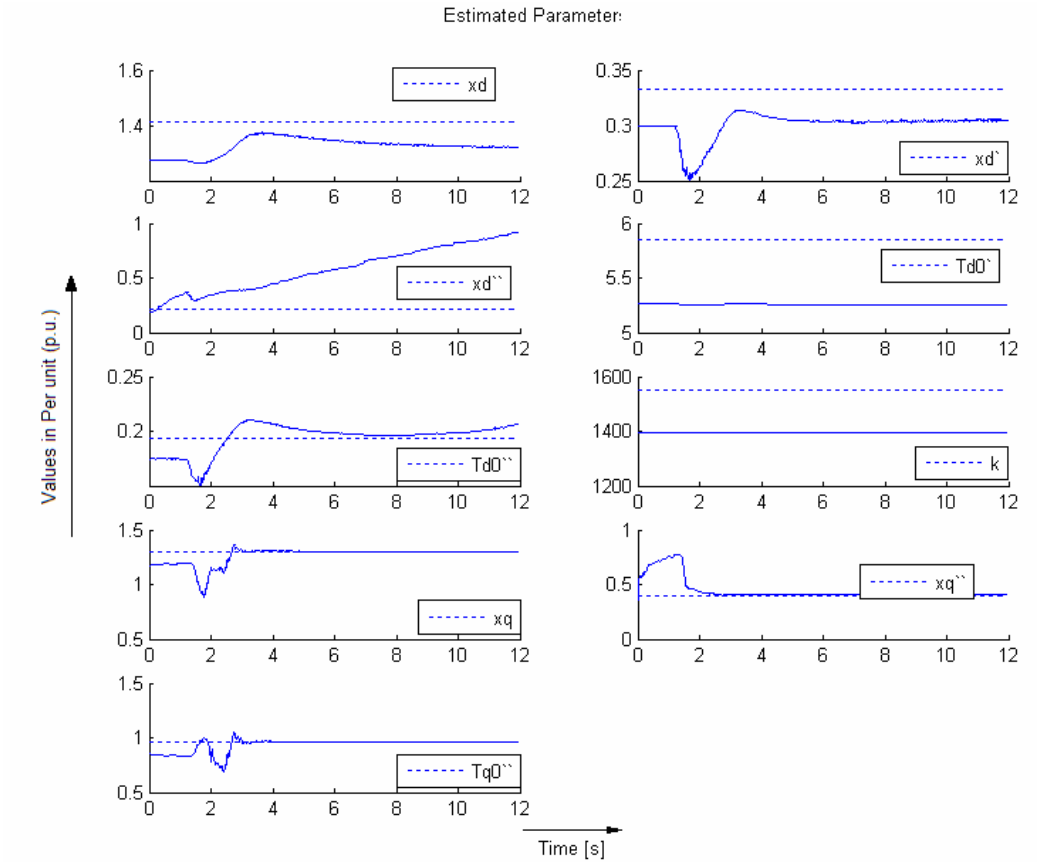

Figure 23. Estimated parameters EKF full order (noise case)

Table 2 Steady state results of Estimation using the EKF: noise free and noisy analysis

Estimation Using Extended Kalman Filter							
Para-meter	***V/ Real***	***Noise free***			***Noisy***		
		V/ initial	***V/ esti-mated***	***error %***	***V/ initial***	***V/ esti-mated***	***error %***
Xd	1.414	1.272	1.362	3.64	1.272	1.309	6.93
Xd'	0.333	0.299	0.333	0.03	0.299	0.298	10.40
Xd''	0.208	0.187	0.207	0.25	0.187	0.895	330.63
Td0'	5.85	5.265	5.274	9.85	5.265	5.255	10.1732
Td0''	0.194	0.174	0.192	0.94	0.174	0.231	18.95
K	1552	1396.8	1396.8	10	1396.8	1396.8	10
Xq	1.302	1.171	1.302	0.00	1.171	1.301	0.05
Xq''	0.396	0.356	0.401	1.34	0.356	0.406	2.65
Tq0''	0.955	0.859	0.940	1.52	0.859	0.945	1.02

In the noise free analysis 7 of 9 parameters estimated converged. The parameters Td0` and K did not change from their initial values. The states and the output estimated were closed to the real values.

The study of the noisy case shows partial convergence for the filter. Parameters 1 to 6 did not converge affected mainly by the nonconvergence of the parameter Xd''. This also results in the nonconvergence of the estimated state Eq". The parameters 7-9 converge to their nominal values, the only difference is the convergence shape of Xd" which has a big overshoot before reaching to the nominal value. However the estimated output remains the same as in the noise free case. In order to modify the EKF a sensitivity analysis is presented in Figure 24.

4.3 CONDITIONING ANALYSIS

Based on the concept presented in Section 1.4, two sensitivity analyses were performed. The first analysis presented in Figure 24 calculates the Jacobian of the output with respect to the estimated states and parameters (Appendix C subfunction "jac_rls.m"). The sensitivity analysis was performed for the noisy case. Is important to see the sensitivity of i_d with respect to $x_q \quad x_q'' \quad T_{qo}''$ are zero and the sensitivity of i_q respect to $x_d \quad x_d' \quad x_d'' \quad T_{d0}' \quad T_{do}'' \quad k$ is also zero. The sensitivity analysis also shows that not all the components are affected by the noise. This means that the Jacobian of the output with respect to Xd" and Xq" contains a big component of noise.

The second analysis presents the conditioning analysis of the EKF. The Hessian matrix is presented in Table 3. The Hessian is a block diagonal matrix with one block associated with the direct axis parameters ($x_d \quad x_d' \quad x_d'' \quad T_{d0}' \quad T_{do}'' \quad k$) and the other with the quadrature axis parameters ($x_q \quad x_q'' \quad T_{qo}''$).Table 4 presents the eigenvalues of the Hessian matrix. The relation between bigger eigenvalue to the smaller is $1.65*10^{12}$ which mean that the Hessian matrix is very ill conditioned.

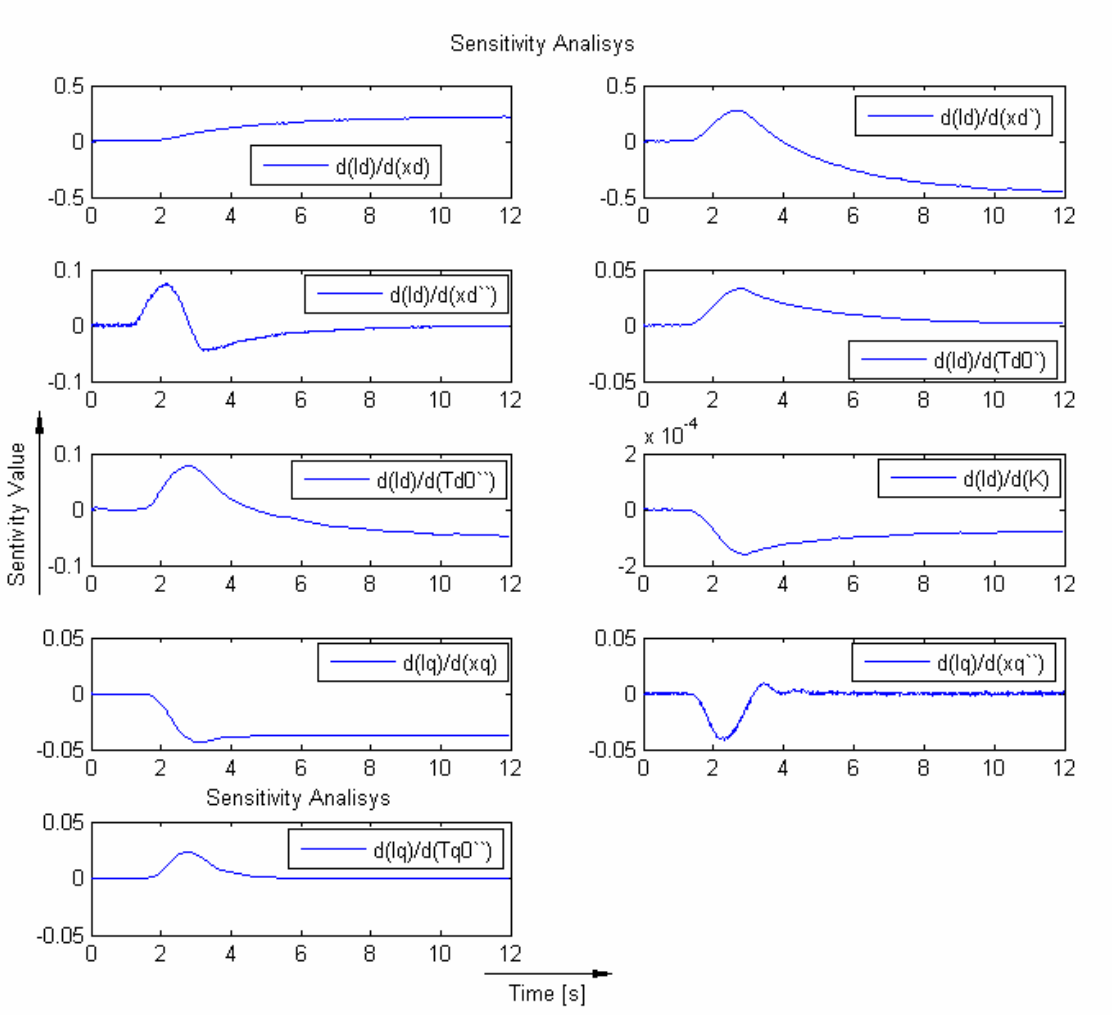

Figure 24. Jacobian of Synchronous Generator Model

Table 3 Hessian matrix of EKF (noisy case)

$$H=\begin{bmatrix} 26.22 & -42.82 & -1.20 & 1.09 & -3.37 & -0.01 & 0 & 0 & 0 \\ -42.82 & 89.54 & 1.82 & -0.19 & 10.05 & 0.02 & 0 & 0 & 0 \\ -1.20 & 1.82 & 0.58 & -0.04 & 0.15 & 0.00 & 0 & 0 & 0 \\ 1.09 & -0.19 & -0.04 & 0.17 & 0.19 & 0.00 & 0 & 0 & 0 \\ -3.57 & 10.05 & 0.15 & 0.19 & 1.40 & 0.00 & 0 & 0 & 0 \\ -0.01 & 0.02 & 0.00 & 0.00 & 0.00 & 9.E\text{-}06 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.20 & 0.06 & -0.11 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.06 & 0.11 & -0.05 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.11 & -0.05 & 0.05 \end{bmatrix}$$

Table 4 Hessian's matrix eigenvalues

Eigenvalues of Hessian Matrix	
λ_1	1.122E+02
λ_2	5.195E+00
λ_3	1.215E+00
λ_4	5.253E-01
λ_5	1.251E-01
λ_6	1.929E-02
λ_7	9.340E-03
λ_8	4.201E-05
λ_9	6.800E-11

Figure 25 and Figure 26 present the covariance condition number as a function of the time for the noise free and the noisy case, respectively. The noise free case presents a condition value which is constant before the transient; during the transient it goes down and after that it starts to increase. The noisy case only differs from the previous after the transient when the condition number remains constant in steady state as presented in (Figure 26) for t>5 [s].

The result shows how the noisy signals affect the Jacobian, the most affected parameters is Xd" and Xq". The sensitivity analysis also shows how the parameter K has a small value and consequently that is the principal reason for its nonconvergence to the real value. The conditioning analysis shows the necessity to improve the EKF to reduce the ill-conditioning problem.

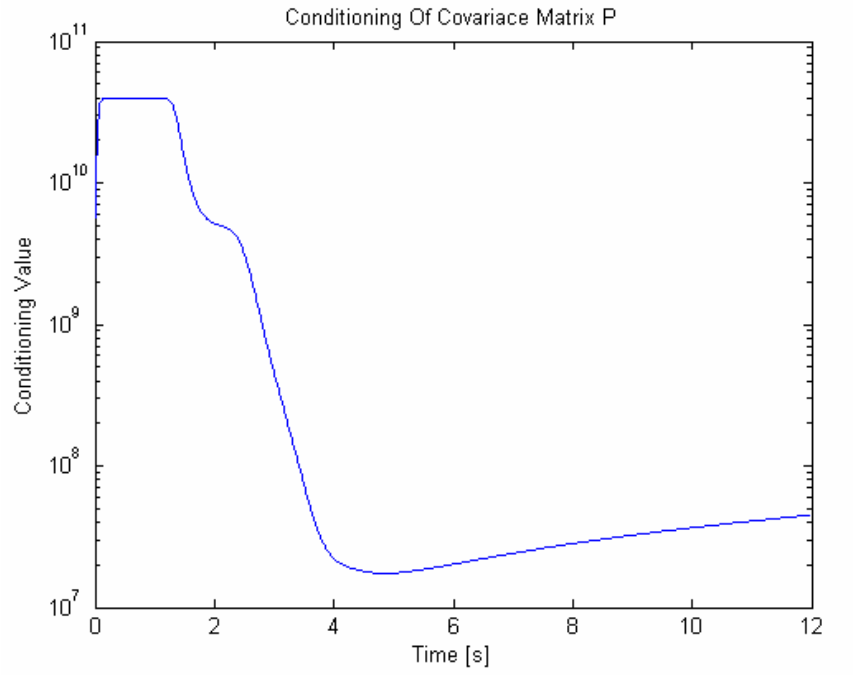

Figure 25. Condition number of error covariance matrix (noise free analysis)

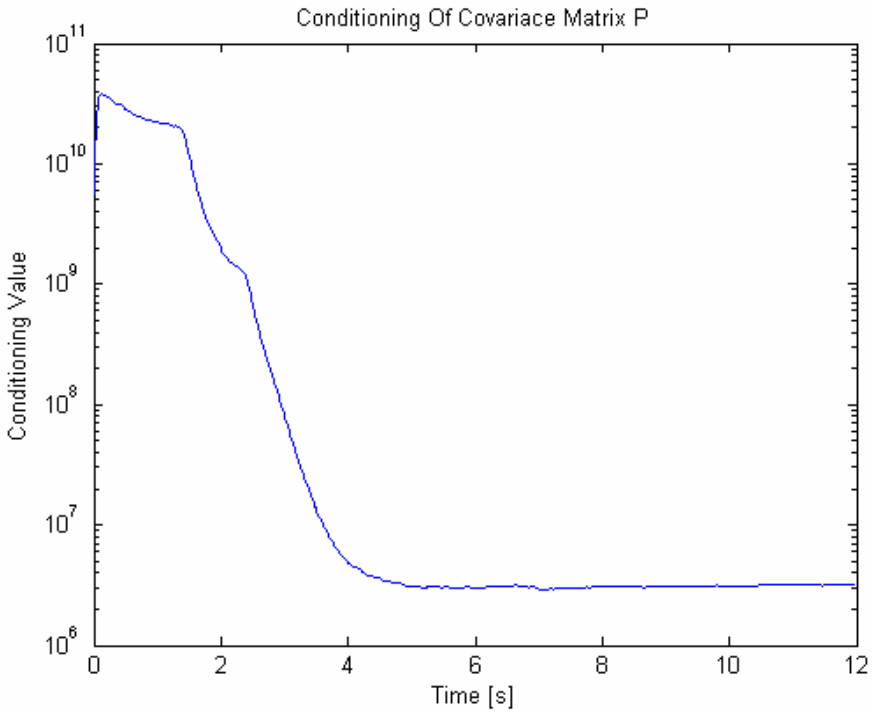

Figure 26. Condition number of error covariance matrix (noisy analysis)

CHAPTER 5 MODIFIED EXTENDED KALMAN FILTER

This chapter presents de modifications applied to the EKF used to increase the performance of the algorithm for parameter identification. In order to improve the convergence of the algorithm, it is necessary to eliminate the ill conditioning of the problem as presented in [Velez-Reyes,1992]. Here we study the use of subset selection to improve the estimation performed.

5.1 SUBSET SELECTION

The subset selection methodology presented in [Velez-Reyes, 1992] and [Jauregui, 2001] bring prior information to the estimation problem to reduce the ill conditioning. The subset selection uses the information of the conditioning analysis to select which parameters must be fixed. In particular, it is shown that for subset selection methodology the algorithm has fast convergence rates and simple structures that are attractive for both off-line and on-line implementations.

When a subset of parameters is fixed the Jacobian has a reduced order and the problem contains a subset of the columns of the full order problem and hence the Hessian is a sub-matrix of the full order Hessian. The objective to reduce the Hessian is to reduce the ill conditioning. Sub set selection requires to search the best combinatorial of p-column of the Jacobian that will result in the best conditioned Hessian. This combinatorial search is feasible in this low dimension problem, in high dimension problems.

The subset selection methodology was applied to the synchronous generator problem with a batch identification algorithm; in [Jauregui, 2001] when they study what is the best selection of parameters to be fixed for the case under study.

The equation (5.1) defines the parameter vector and assigns an index number to each parameter.

$$\mathbf{P} = [\underset{\downarrow}{x_d} \quad \underset{\downarrow}{x_d^{'}} \quad \underset{\downarrow}{x_d^{''}} \quad \underset{\downarrow}{T_{d0}^{'}} \quad \underset{\downarrow}{T_{do}^{''}} \quad \underset{\downarrow}{k} \quad \underset{\downarrow}{x_q} \quad \underset{\downarrow}{x_q^{''}} \quad \underset{\downarrow}{T_{qo}^{''}}]^T \qquad (5.1)$$

$$1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9$$

The parameters fixed are two; this number is based on the conditioning analysis where two of the Hessian's eigenvalues have a smaller compared with the others (λ_8=4.201E-05 λ_9=6.800E-11). Table 5 summarizes the result of computation of the condition number of the reduced order Hessian for different sets of parameters for batch estimation.

Table 5 Condition Number of Subset selection batch technique

Combinations	Condition number	Combinations	Condition number
8 5 3 2 1 4 6	1.34E+14	7 9 5 3 2 4 6	5.40E+08
9 5 3 2 1 4 6	1.34E+14	7 9 5 3 2 1 6	2.51E+06
9 8 3 2 1 4 6	2.97E+09	7 9 5 3 2 1 4	2.41E+07
9 8 5 2 1 4 6	6.38E+09	7 9 8 2 1 4 6	2.82E+07
9 8 5 3 1 4 6	2.57E+09	7 9 8 3 1 4 6	8.15E+07
9 8 5 3 2 4 6	1.34E+09	7 9 8 3 2 4 6	1.33E+07
9 8 5 3 2 1 6	5.40E+08	7 9 8 3 2 1 6	7.31E+04
9 8 5 3 2 1 4	2.51E+06	7 9 8 3 2 1 4	3.58E+08
7 5 3 2 1 4 6	1.34E+14	7 9 8 5 1 4 6	3.58E+08
7 8 3 2 1 4 6	2.97E+09	7 9 8 5 2 4 6	1.99E+07
7 8 5 2 1 4 6	6.38E+09	7 9 8 5 2 1 6	1.03E+07
7 8 5 3 1 4 6	2.57E+09	7 9 8 5 2 1 4	4.37E+04
7 8 5 3 2 4 6	1.34E+09	7 9 8 5 2 1 4	9.23E+04
7 8 5 3 2 1 6	2.51E+06	7 9 8 5 3 1 6	6.46E+05
7 8 5 3 2 1 4	2.97E+09	7 9 8 5 3 1 4	3.20E+03
7 9 3 2 1 4 6	6.38E+09	7 9 8 5 3 2 6	3.39E+05
7 9 5 2 1 4 6	2.57E+09	7 9 8 5 3 2 4	4.59E+03
7 9 5 3 1 4 6	1.34E+09	7 9 8 5 3 2 1	5.17E+03

The parameters fixed are two; this number is based on the conditioning analysis where two of the Hessian's eigenvalues have a smaller compared with the others (λ_8=4.201E-05 λ_9=6.800E-11). Table 5 summarizes the result of computation of the condition number of the reduced order Hessian for different sets of parameters for batch estimation.

Table 5 shows that fixing Xd and Td0' or Td0' and K result in highest improvement in condition number (ej. With lowest condition number)

Table 6 Steady state Results of Subset Selection applied to Gauss Newton method

Subset Selection applied to the Gauss Newton Method						
Para-meter	***V/ Real***	***Full order estimation***	***Estimation of 7 parameters K and Xd fixed***		***Estimation of 7 parameters K and Td0' fixed***	
		V/ esti-mated	***V/ esti-mated***	***V/ esti-mated***	***V/ esti-mated***	***error %***
Xd	1.414	22.07	-	-	1.414	0.0384
Xd'	0.333	45.109	0.333	0.28	0.334	0.118
Xd''	0.208	28.393	0.235	13.885	0.196	5.664
Td0'	5.85	187.048	5.881	0.539	-	-
Tdo''	0.194	44.148	0.186	3.67	0.182	6.594
K	1552	145.345	-	-	-	-
Xq	1.302	11.94	1.303	0.13	1.302	0.007
Xq''	0.396	20.474	0.394	0.538	0.393	0.04
Tq0''	0.955	14.173	0.957	0.212	0.955	0.0124

Table 6 shows the improvement to batch Gauss-Newton by the application of subset selection methodology. The same methodology was applied to the EKF in order to improve the algorithm. The first approach was to fix the parameters presented in Table 6. The simulations of the EKF with subset selection for the noisy case shows the non convergence of the estimators, principally affected by the parameter Xd" and the state variable Eq".

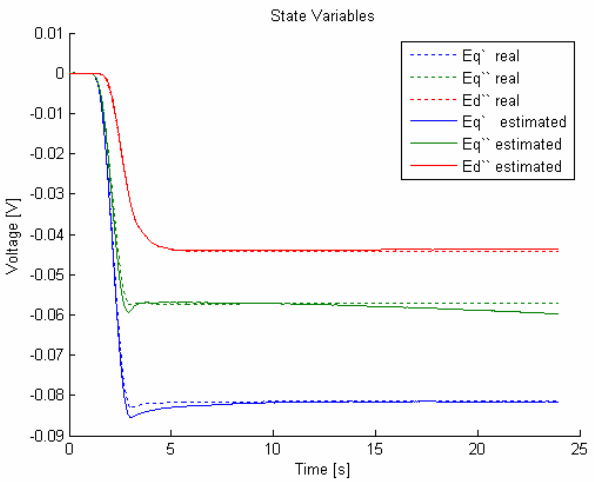

Figure 27. Estimated sates with Xd and K fixed for subset selection EKF

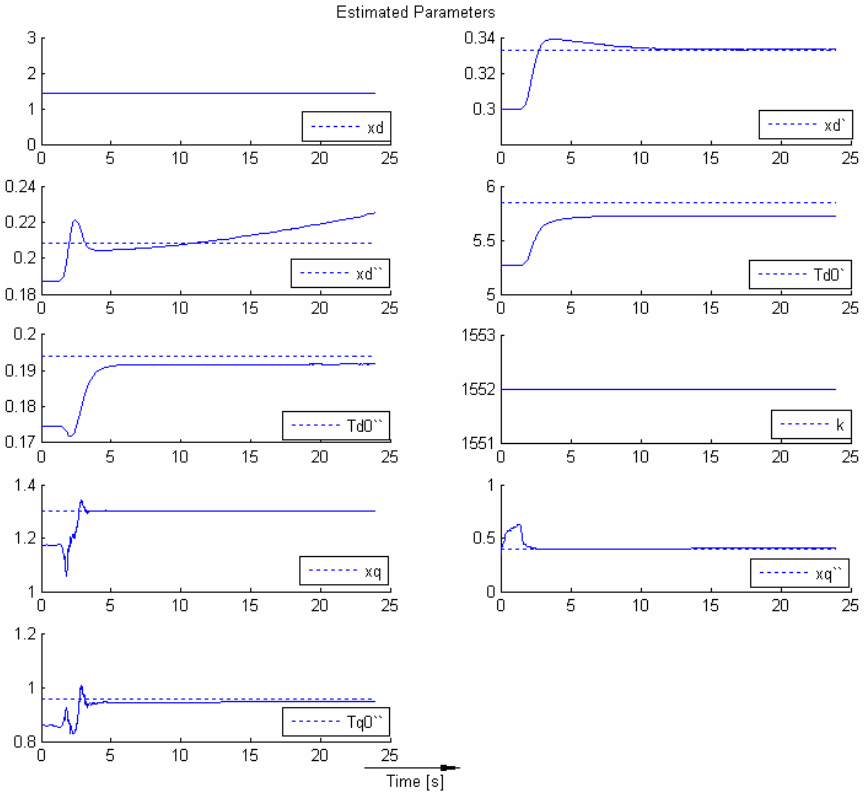

Figure 28. Estimated parameters with Xd and K fixed for subset selection EKF

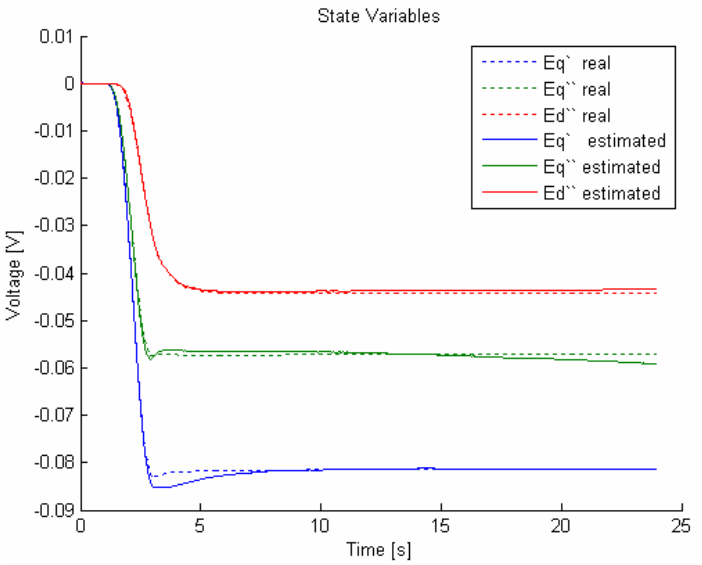

Figure 29. Estimated sates with Td0' and K fixed for subset selection EKF

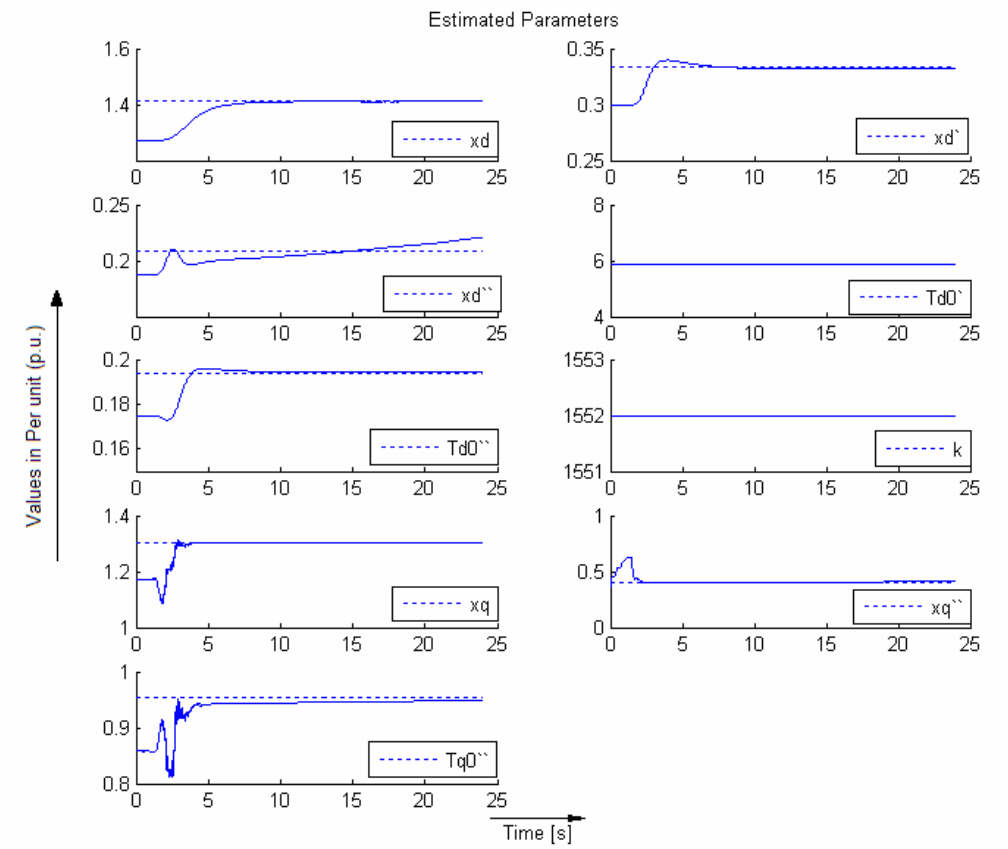

Figure 30. Estimated parameters with Td0' and K fixed for subset selection EKF

Fixing 2 parameters does not work for the recursive approach as it does for batch case. Based on the previous results is possible to identify additional parameters to be fixed. For the noise free case (Figure 31 to Figure 33) was fixed K and for the noisy case (Figure 34 to Figure 36) was fixed Xd" and K. Table 7 summarizes the results of the application of the subset selection methodology to the EKF. The results show a high improvement in the convergence in the process of identification of the states and also the parameters. The noisy causes with noise in the convergence waves for the last 3 parameters

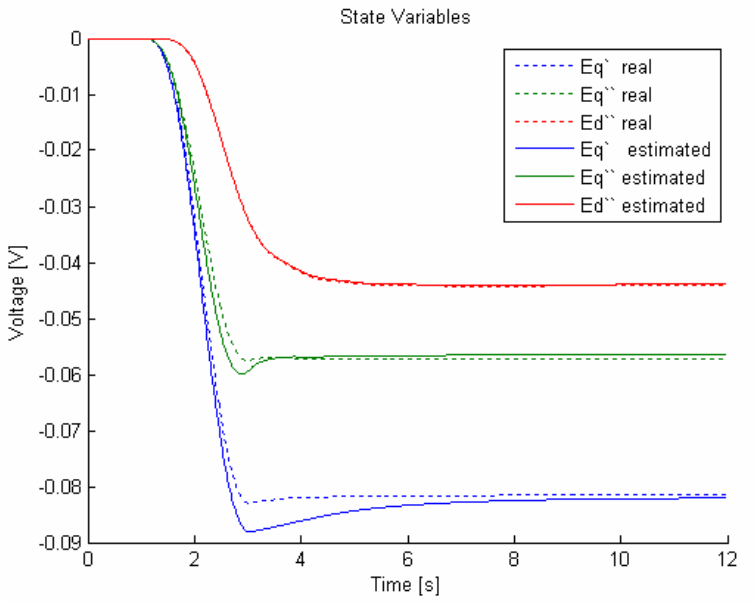

Figure 31. Estimated sates with K fixed SS-EKF (noise free)

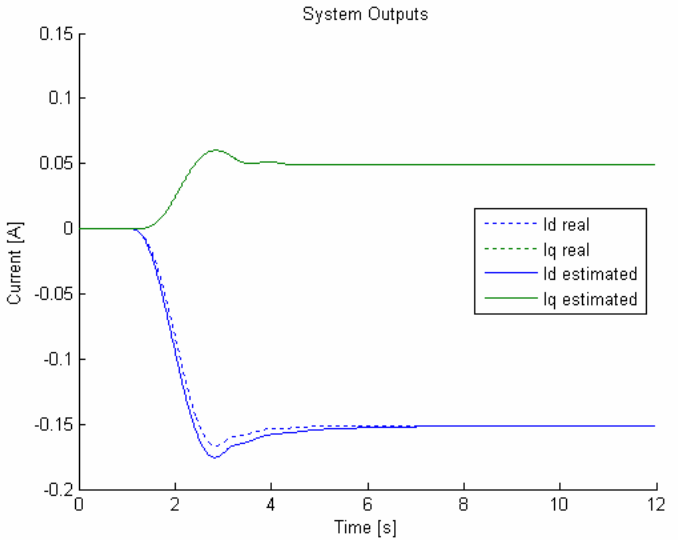

Figure 32. Estimated outputs with K fixed SS-EKF (noise free)

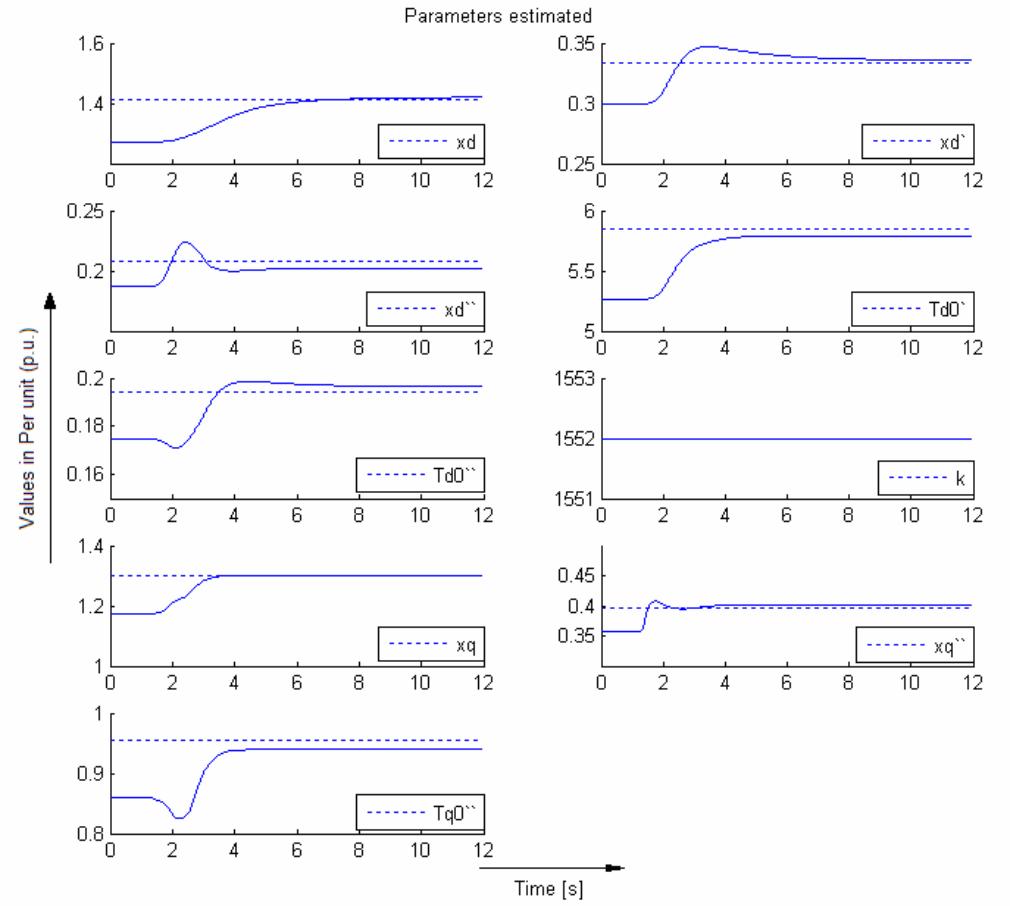

Figure 33. Estimated parameters with K fixed SS-EKF (noise free)

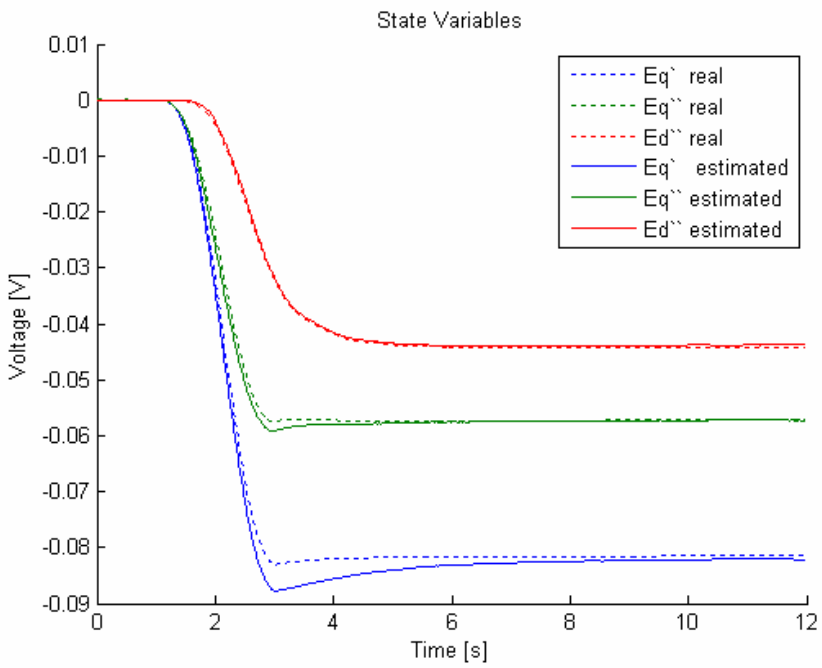

Figure 34. Estimated states with Xd", and K fixed SS-EKF (noisy case)

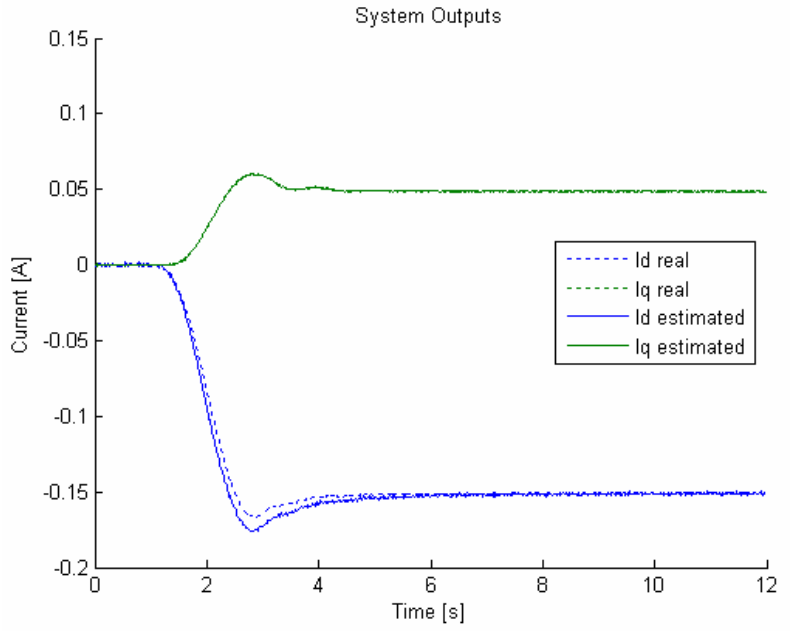

Figure 35. Estimated outputs with Xd" and K fixed SS-EKF (noisy case)

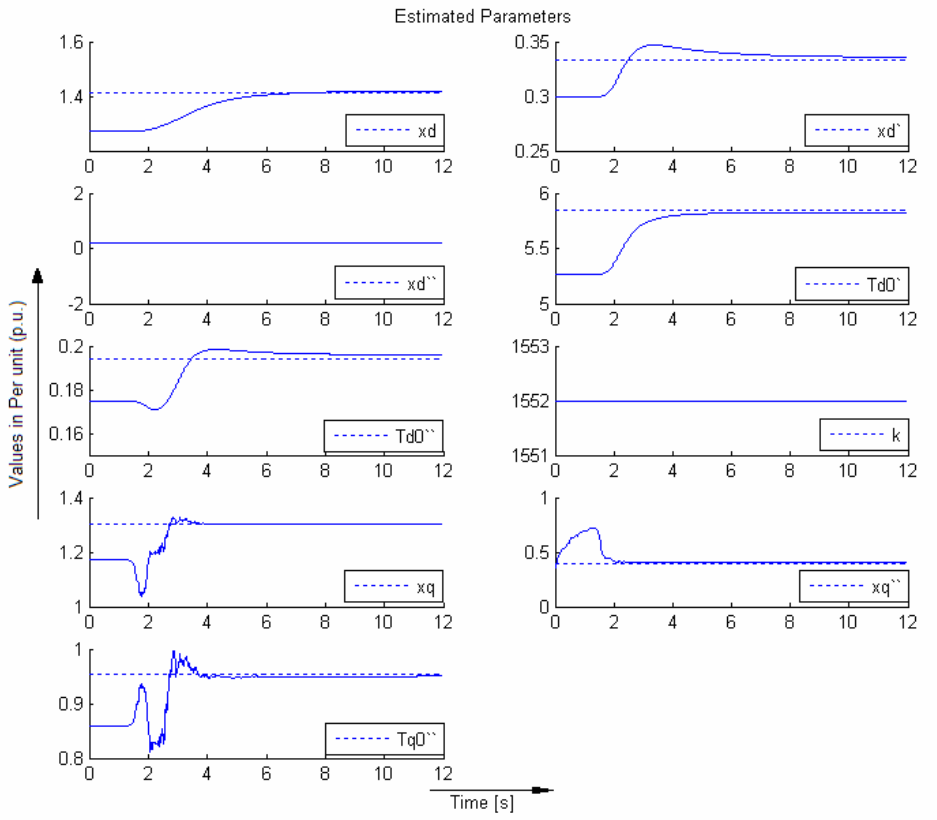

Figure 36. Estimated parameters with Xd" and K fixed SS-EKF (noisy case)

Table 7 Steady state results from estimation using the subset selection- EKF

Estimation Using subset selection Extended Kalman Filter							
Para-meter	***V/ Real***	***Noise Free***			***Noisy***		
		V/ initial	***V/ esti-mated***	***error %***	***V/ initial***	***V/ esti-mated***	***error %***
Xd	1.414	1.272	1.420	0.42	1.272	1.419	0.45
Xd'	0.333	0.299	0.335	0.83	0.299	0.335	0.86
Xd''	0.208	0.187	0.202	2.72	-	-	-
Td0'	5.85	5.265	5.792	0.98	5.265	5.820	0.50
Td0''	0.194	0.174	0.196	1.28	0.174	0.196	1.02
K	1552	-	-	-	-	-	-
Xq	1.302	1.171	1.302	0.00	1.171	1.302	0.02
Xq''	0.396	0.356	0.401	1.26	0.356	0.485	3.19
Tq0''	0.955	0.859	0.940	1.54	0.859	0.954	0.10

5.2 ITERATED EXTENDED KALMAN FILTER (IEKF)

The Iterated Extended Kalman Filter (IEKF) presented in [Gelb, 1974] improves the convergence of the traditional EKF by iterating ever the estimate in more than one time every sample. The process requires that the covariance matrix, the Kalman Gain and the extended state are updated by the linearization about the most resent estimate. The i^{th} iteration (i=0, 1, 2 .. N) of x_i is defined by $x_{k,i}$, with $x_{k,0}= x_k$ (the same notation can be used for the Kalman gain and for the covariance matrix). Taking the first order approximation of $h^e(x^e_{k,i})$:

$$h^e(x^e_{k,i}) = h^e(\hat{x}^e_{k|i}(+)) + C_k(\hat{x}^e_{k|i}(+))\left[\hat{x}^e_K - \hat{x}^e_{k,i}\right] \quad (5.2)$$

is possible to redefine (4.42)-(4.46). The equation (4.42) is used in the first iteration and (4.43) is used only in the final iteration (because they are predictor of the state and the covariance matrix). The equations below summarize the IEKF algorithm. However the notation differs from the used for EKF in order to include the iteration process.

The iterations are associated with the subscript i (i=0, 1, 2 ...N), and the time samples with the sub scrip k. For the first iteration i=0;

$$\hat{x}_{k+1,i} = f(\hat{x}_{k,N}, u_k) \quad (5.3)$$

$$x^e_{k+1,i} = \begin{bmatrix} x_{k+1,i} \\ p_k \end{bmatrix} \quad (5.4)$$

If the iteration is $0 < i \leq N$

$$K_i = P_{k,N} C_i^T (W + C_i P_{k,N} C_i^T)^{-1} \quad (5.5)$$

$$P_{k+1|i} = P_{k+1|i} - K_i C_i P_{k+1|i} \quad (5.6)$$

$$\hat{x}^e_{k+1,i} = \hat{x}^e_{k,N} + K_i\left[z_{k+1} - h\ (\hat{x}_{k+1,i}) - C_i\left[\hat{x}^e_{k,N} - \hat{x}^e_{k+1,i}\right]\right] \quad (5.7)$$

for the last iteration i=N:

$$P_{k+1,N} = A_N P_{k+1,N} A_N^T + V^e \quad (5.8)$$

Figure 37 summarizes the algorithm of the Iterated Extended Kalman Filter algorithm. The principal advantage of the IEKF is the performance improvement by error reduction caused by the system nonlinearities because of the improved reference trajectories incorporated into the estimates. However this process increments the number of calculations required to estimate the parameters each sample. The number of iterations can be selected as the trade off between the value to reach the improvement required over the EKF and the added time required to estimate the parameters.

The Iterated Extended Kalman Filter (IEKF) was applied with the Subset Selection methodology to the synchronous generator model. Two simulations were performed; noise free and noisy.

In the noise free analysis for the IEKF one parameter was fixed such as the EKF. The performance of the IEKF is better for the estimation of the states (see Figure 38 and Figure 31) and also for the output (see Figure 39 and Figure 32). These figures show estimated states closer to the real values especially during transients. However the performance related to the parameter estimation is very similar to the EKF (Figure 40 and Figure 33).

The noisy analysis presents the similar results to the noise free case. Figure 41 to Figure 43 present the results of the identification of states, outputs and parameters, respectively. Table 8 summarizes the data of IEKF with Subset Selection for noisy and noise free analysis.

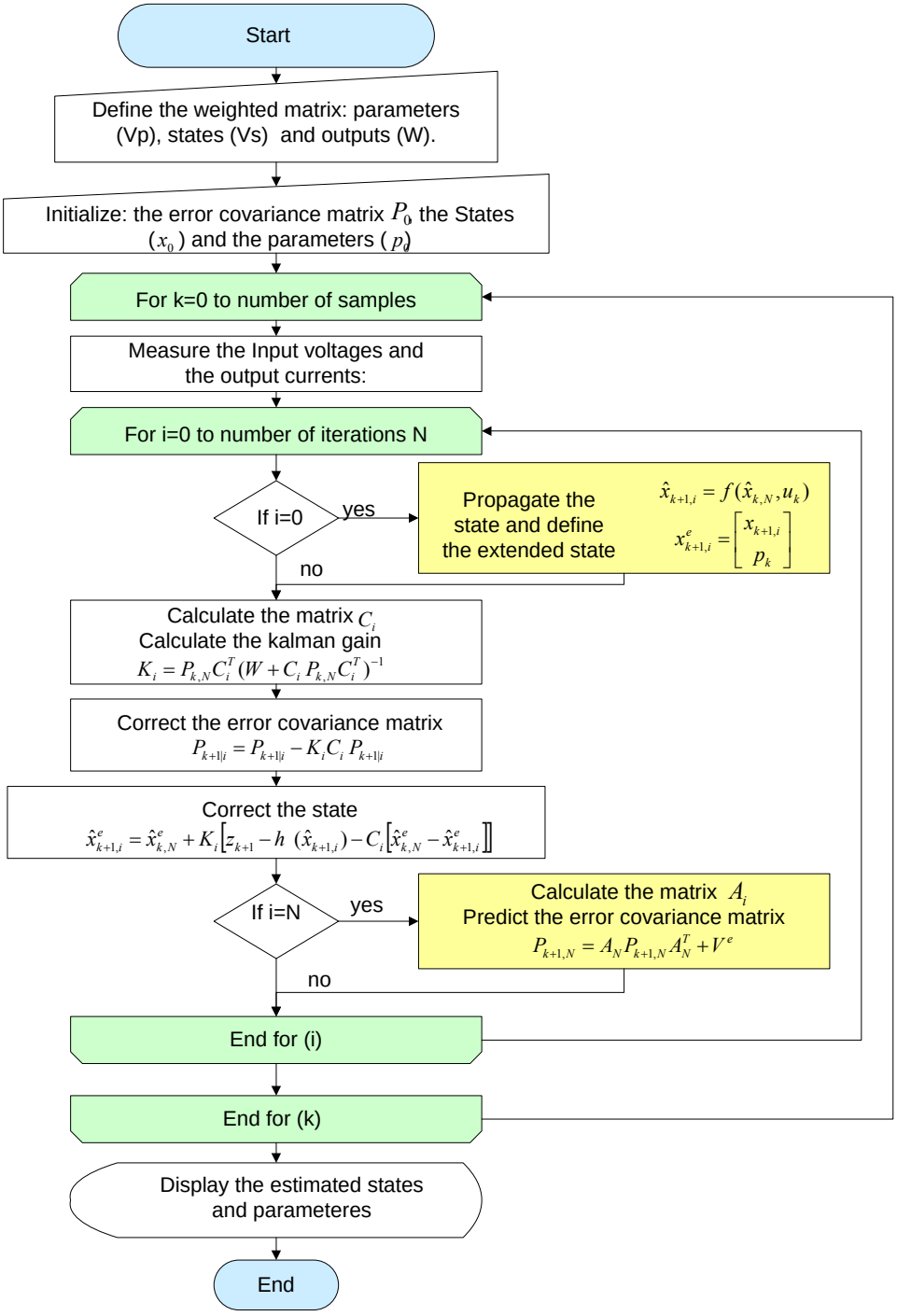

Figure 37. Iterated Extended Kalman Filter algorithm

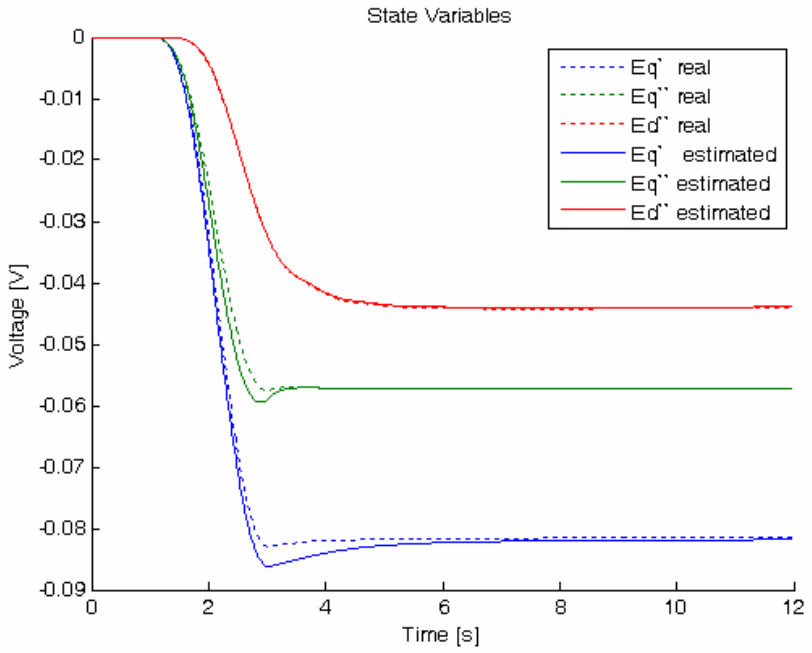

Figure 38. Estimated states with K fixed SS-IEKF (noise free)

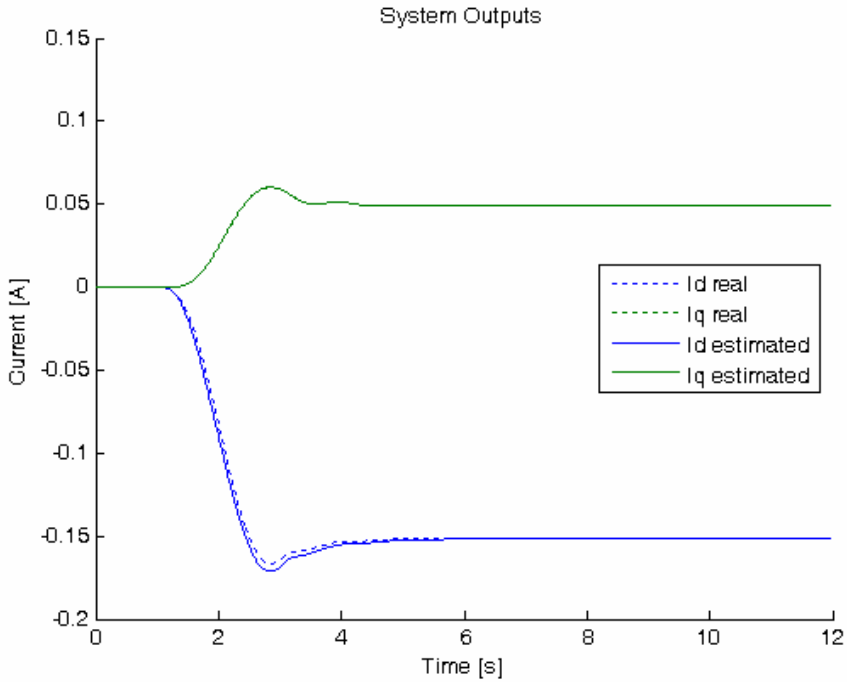

Figure 39. Estimated outputs with K fixed SS-IEKF (noise free)

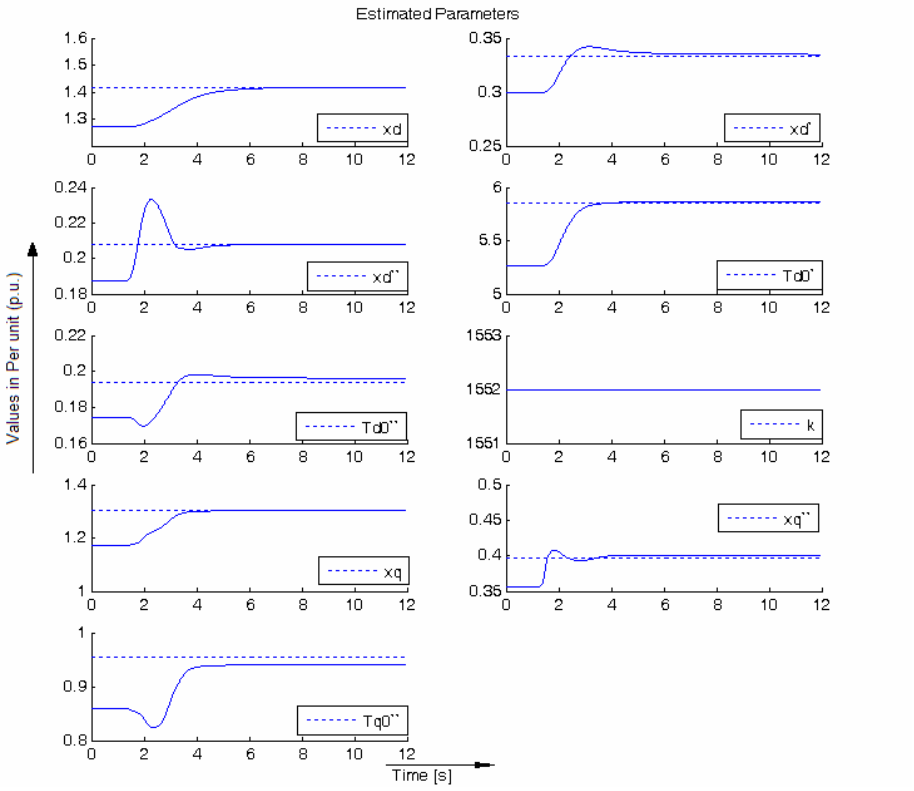

Figure 40. Estimated parameters with K fixed SS-IEKF (noise free)

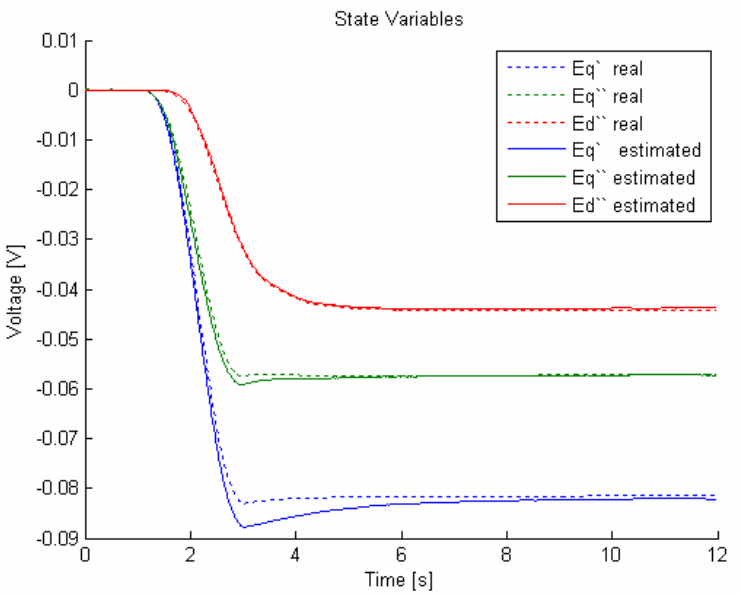

Figure 41. Estimated states with Xd" and K fixed SS-IEKF (noisy case)

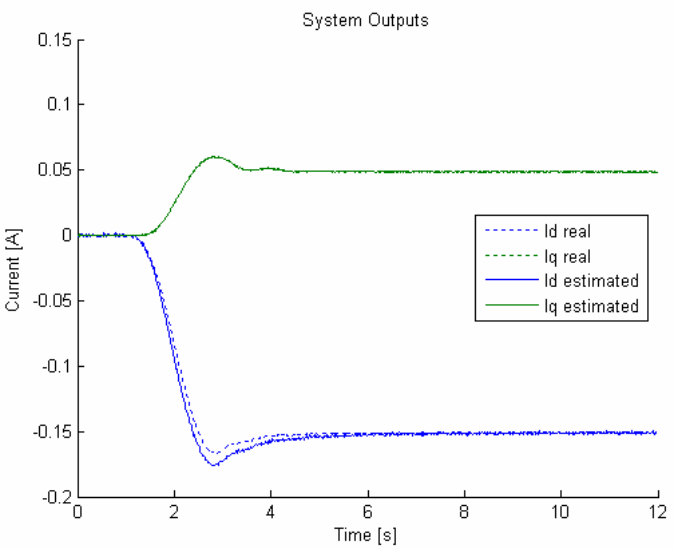

Figure 42. Estimated outputs with Xd", Tq0" and K fixed SS-IEKF (noisy case)

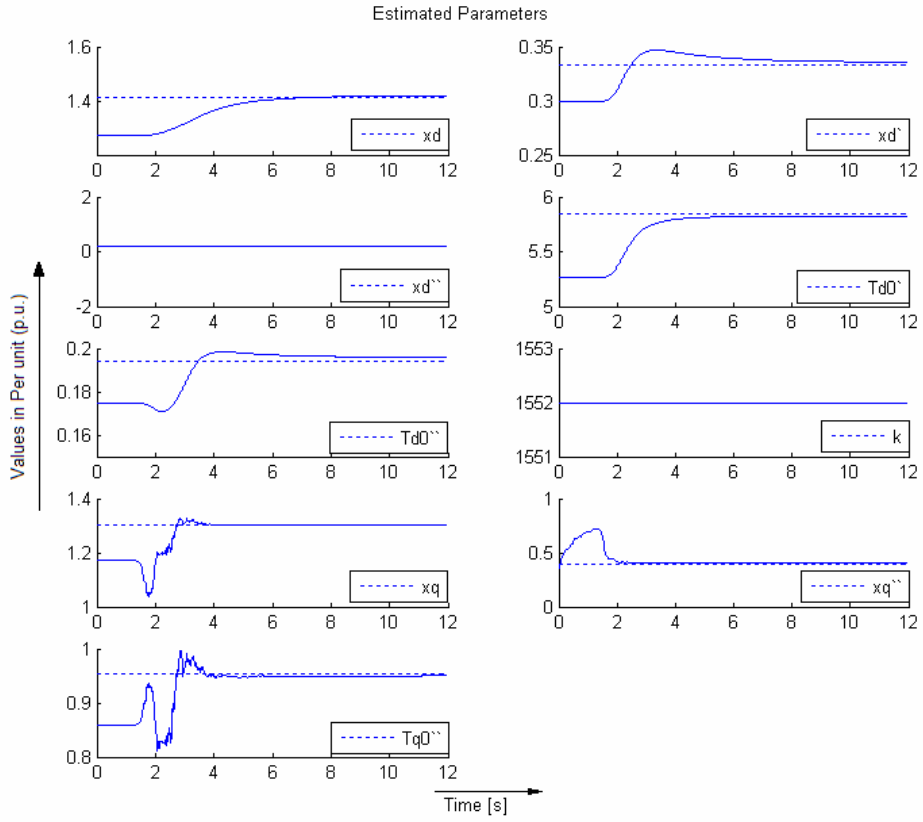

Figure 43. Estimated parameters with Xd" and K fixed SS-IEKF (noisy case)

Table 8 Steady state results of Estimation using the subset selection- IEKF

Estimation Using subset selection Iterated Extended Kalman Filter							
Para-meter	***V/ Real***	***Noise free***			***Noisy case***		
		V/ initial	***V/ esti-mated***	***error %***	***V/ initial***	***V/ esti-mated***	***error %***
Xd	1.414	1.272	1.418	0.29	1.272	1.411	0.23
Xd'	0.333	0.299	0.335	0.56	0.299	0.332	0.22
Xd''	0.208	0.187	0.208	0.05	-	-	-
Td0'	5.85	5.265	5.861	0.018	5.265	5.897	0.81
Td0''	0.194	0.174	0.196	1.14	0.174	0.195	0.35
K	1552	-	-	-	-	-	-
Xq	1.302	1.171	1.302	0.00	1.171	1.302	0.00
Xq''	0.396	0.356	0.401	1.16	0.356	0.403	1.84
Tq0''	0.955	0.859	0.941	1.52	0.859	0.947	0.80

CHAPTER 6 CONCLUSIONS AND RECOMMENDATION

6.1 CONCLUSIONS

In this thesis, we presented the Extended Kalman Filter applied to the synchronous generator parameter identification problem. A simplified model was considered, this model with the selected synchronous generator parameters resulted an ill conditioned problem. Ill condinitioning resulted in non convergence of the Kalman Filter. To deal with ill- conditioning, various methodologies were proposed in order to improve the convergence of the EKF.

Chapter 3 analyzed batch estimation; the methodologies studied were Gauss-Newton and Levenberg-Marquardt. The Gauss-Newton was studied to give information for design of the recursive estimation algorithm. Levenberg-Marquardt method was analyzed as the regularization alternative for batch identification; giving improvements over the Gauss-Newton methodology, specifically under the noisy simulations were the Levenberg-Marquardt avoids the non convergence problems presented in the Gauss-Newton.

Chapter 4 presented the EKF algorithm. The simulation performed with the EKF showed the convergence for the algorithm for the noise free case, and the partial convergence for the noisy simulation. These simulations showed how the parameter K does not change its initial value. Also the results showed a decoupling between the Direct axis and Quadrature axis parameters. The conditioning analysis for this model present how the noisy input-output signals partially affect the Jacobian especially the components related to the Xd" and Xq". The condition analysis gives sufficient information to conclude that there is decoupling between the Quadrature and Direct axis component showed initially by the EKF (the Hessian is a two blocks diagonal matrix).

Chapter 5 implemented the sub-set selection with the EKF. To select the parameters to be fixed, the results of the EKF and the conditioning analysis were used. The fixed parameters were: K for the noise free simulation and K and Xd" for the noisy case. The

result shows a big improvement with the application of this methodology, because under this condition the non fixed parameters converged to the nominal values and the state estimates were close to those obtained with the nominal parameters; the estimation under noisy conditions only converges for the EKF with subset selection.

The Iterated extended Kalman Filter was also studied. Full parameter estimation has the same problems as in the EKF. The simulation performed include the application of the subset selection methodology, the results showed how the IEKF estimates have better state estimation accuracy (the maximum error is 3% for the IEKF versus 8% for the EKF). However, the performance related to the parameter estimated is worst (during the transient the maximum deviation from the real value for the IEKF is 25% versus the 15% for the EKF). The additional calculus of the covariance, Kalman gain and the new estimates for additional iterations make it impractical the application of the IEKF; this conclusion is based on the additional computations required for a little improvement in the steady state results (only for 1%).

The main results obtained in this thesis can be summarized as follows:

- The Extended Kalman Filter combined with subset selection is an accurate method for recursive parameter identification.
- The Subset selection is the best methodology to use with the EKF for ill conditioned problems however it requires a priori values of some parameters and a sensitivity analysis.
    - With the EKF and noisy case the method does not converge, however for he same case but with subset selection is possible to estimate 7 of 9 parameters.
- The Iterated Extended Kalman Filter is an alternative for recursive parameter identification; it has better convergence for estimating the states but worse for parameter estimation.
    - For the noisy case simulation, the estimated parameters have approximately the same errors for the EKF and for the IEKF. However the IEKF requires the

calculus of the covariance, the Kalman gain and the new estimates for additional iterations within a sampling interval compared with the traditional EKF.

- The UD factorization implementation of the EKF was studied but it did not improve the results versus the traditional EKF equations.

6.2 FUTURE WORK

In this thesis, we showed a recursive parameter identification technique for synchronous generators, the ill conditioning problem was presented and methodologies were proposed to reduce it. Based on the literature where presented how nonlinear models are accurately to the real system; future work challenge would be to use a nonlinear model to design more accurately robust nonlinear estimator.

Other alternatives will consider extensions of batch regularization and variable dimension methods to recursive parameter estimation [Jauregui, 2001] and [Huaman, 2005] .

Recommendations include to extend the application of the proposed EKF with subset selection to the estimation of parameters, speed and load of other machines like: induction motors and synchronous motor. The application of EKF-SS allows creating methodologies for adaptive control and failure detection in those machines [Aquino, 2006].

REFERENCES

[Abido, 1997] Abido, M.A.; Abdel-Magid, Y.L., "On-line identification of synchronous machines using radial basis function neural networks," on Power Systems, IEEE Transactions , vol.12, no.4pp.1500-1506, Nov 1997.

[Aquino, 2006] Aquino A. Gray Box Modeling Of Electric Drives Using Radial Basis Functions: An Experimental Case , M. S. thesis, University Of Puerto Rio at Mayagüez, Mayagüez, Puerto Rico, 2005.

[Burth et al, 1999] Burth, M.; Verghese, G.C.; Velez-Reyes, M., "Subset selection for improved parameter estimation in on-line identification of a synchronous generator," IEEE Transactions on Power Systems , vol.14, no.1pp.218-225, Feb 1999

[Gelb, 1974] Arthur Gelb, ed., Applied Optimal Estimation, MIT Press, Cambridge, MA pp 180-191, 1974.

[Gurnason et al, 1996] Gunnarsson, S., "Combining tracking and regularization in recursive least squares identification," Proceedings of the 35th IEEE Decision and Control, 1996, vol.3, no.pp.2551-2552 vol.3, 11-13 Dec 1996.

[Huaman, 2005] L. Huaman, A Variable Dimension Gauss-Newton Method For Ill-Conditioned Parameter Estimation with Application to a Synchronous Generator, M. S. thesis, University Of Puerto Rio at Mayagüez, Mayagüez, Puerto Rico, 2005.

[IEEE guide, 1991] IEEE guide for synchronous generator modeling practices in stability analyses, IEEE Std 1110-1991 , vol., no.pp.-, Nov 1991.

[IEEE Standard, 1995] IEEE Standard 115-1995 IEEE Guide: Test Procedures for Synchronous Machines, Part I—Acceptance and Performance Testing, Part II—Test Procedures and Parameter Determination for Dynamic Analysis.

[Jauregui, 2001] L. Jáuregui, Ill-condition In Parameter Estimation with application to synchronous generator, M. S. thesis,University Of Puerto Rio at Mayagüez, Mayagüez, Puerto Rico, 2001.

[Jin-Cheng et al, 1994] Jin-Cheng Wang; Hsiao-Dong Chiang; Chiang-Tsung Huang; Yung-Tien Chen; Chung-Liang Chang; Chiew-Yann Chiou, "On-line measurement-based

model parameter estimation for synchronous generators: solution algorithm and numerical studies," IEEE Transactions on Energy Conversion , vol.9, no.2pp.337-343, Jun 1994.

[Jingmin et al, 2001] Jingmin Xin; Sano, A., "MSE-based regularization approach to direction estimation of coherent narrowband signals using linear prediction," IEEE Transactions on Signal Processing, [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on] , vol.49, no.11pp.2481-2497, Nov 2001

[Jung, 1999] L. Jung, System Identification, Second Edition, pp 361-398, 1999.

[Kundert, 1986] K. S. Kundert, "Sparse Matrix Technique in Circuit Analysis, Simulation and Design", A. E. Ruehli, Ed. Amsterdam, the Netherlands: Elsevier, 1986.

[Kundur, 1993] P. Kundur, Power System Stability and Control. Mc Graw Hill, 1994.

[Maybeck,1979] P.S. Maybeck, Stochastic Models, Estimation and Control (Vol. 1), Academic Press, 1979. pp 369-405.

[Narendra et al, 1989] K Narendra and A. Ammaswamy. Stable adaptive Systems. pp 141-144. New Jersey 1989.

[Ng et al, 2000] Ng, S.W.; Lee, Y.S. "Variable Dimension Newton-Raphson Method" IEEE Transactions Circuits and Systems I: Fundamental Theory and Applications, Vol.47, Iss.6, Jun 2000 Pages:809-817.

[Ngia et al, 2000] Ngia, L.S.H., "Separable nonlinear least-squares methods for on-line estimation of neural nets Hammerstein models," Proceedings of the 2000 IEEE Signal Processing Society Workshop Neural Networks for Signal Processing X, 2000. vol.1, no.pp.65-74 vol.1, 2000.

[Polak, 2001] Polak A. "Indirect Measurements: Combining Parameter Selection with Ridge Regression" Measurement Science and Technology, Volume 12, Number 3, 2001, pp. 278-287.

[Robet el al, 1995] Robet, P.P.; Gautier, M.; Bergmann, C., "Estimation of the synchronous drive electrical parameters" Proceedings of the 4th IEEE Conference on Control Applications, 1995. vol., no.pp.514-519, 28-29 Sep 1995

[Telford et al, 2003] Telford, D.; Dunnigan, M.W.; Williams, B.W., "Online identification of induction machine electrical parameters for vector control loop tuning," IEEE Transactions on Industrial Electronics, vol.50, no.2pp. 253- 261, Apr 2003.

[Vélez-Reyes,1992] Vélez-Reyes M., 1992, Decomposed Algorithm for Parameter Estimation. Ph. D.Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 221 pp.

[Vahidi et al, 2003] A. Vahidi, A. Stefanopoulou, H. Peng. “Recursive Least Squares with Forgetting for Online Estimation of Vehicle Mass and Road Grade: Theory and Experiments,” Vehicle System Dynamics, Jan 16, 2003.

[Walter et al, 1994] E. Walter, and L Pronzato. Identification of Parametric Models. pp 111-113, Great Britain 1994.

[Wang et al, 1995] Wang, J.-C.; Chiang, H.-D.; Huang, C.-T.; Chen, Y.-T., "Identification of synchronous generator saturation models based on on-line digital measurements," IEE Proceedings- Generation, Transmission and Distribution, vol.142, no.3 pp.225-232, May 1995.

[Zhao et al, 1995] Z.M Zhao, F.S. Zheng, J.D. Gao and L.Y. Xu, “A Dynamic on-Line Parameter identification and Full-scale Sysem Experimental Verification for Large Synchronous Machines ”, IEEE Trans. Energy Conversion, vol EC10, N3, pp 392-398 September 1995.

APPENDIX A
JACOBIAN AND KALMAN MATRICES STRUCTURE

A.1 Jacobian calculation

In the process of sensitivity analysis was necessary to calculate the Jacobian. First is necessary to define the linear system in the form:

$$\begin{aligned} \dot{\overrightarrow{x(t)}} &= A(\mathbf{p})x(t) + B(\mathbf{p})u(t) \\ y(t) &= C(\mathbf{p})x(t) + D(\mathbf{p})u(t) \end{aligned} \qquad (A.1)$$

Where

$$A(\mathbf{p}) = \begin{bmatrix} \frac{1}{T_{do}} & -\left(\frac{xd - x'd}{T_{do}^{'} x_{d}^{''}}\right) & 0 \\ \left(\frac{1}{T_{do}^{''}} - \frac{1}{T_{do}^{'}}\right) & \left(\frac{1}{T_{do}^{''}} + \frac{xd - x'd}{T_{do}^{'} x''d} + \frac{x'd - x''d}{T_{do}^{''} x''d}\right) & 0 \\ 0 & 0 & \frac{xq}{T_{qo}^{''} x_{q}^{''}} \end{bmatrix} \qquad (A.2)$$

$$B(\mathbf{p}) = \begin{bmatrix} 0 & \frac{xd - x'd}{T_{do}^{'} x_{d}^{''}} & \frac{k}{T_{do}^{'}} \\ 0 & \left(\frac{xd - x'd}{T_{do}^{'} x''d} + \frac{x'd - x''d}{T_{do}^{''} x''d}\right) & \frac{k}{T_{do}^{'}} \\ \frac{xq - x''d}{T_{qo}^{''} x_{q}^{''}} & 0 & 0 \end{bmatrix} \qquad (A.3)$$

$$C(\mathbf{p}) = \begin{bmatrix} 0 & \frac{1}{x_{d}^{''}} & 0 \\ 0 & 0 & \frac{1}{x_{q}^{''}} \end{bmatrix} \qquad (A.4)$$

$$D(\boldsymbol{\alpha}) = \begin{bmatrix} 0 & \dfrac{1}{x''_d} & 0 \\ \dfrac{1}{x''_q} & 0 & 0 \end{bmatrix} \qquad (A.5)$$

And $\mathbf{p} = [x_d \quad x'_d \quad x''_d \quad T'_{d0} \quad T''_{do} \quad k \quad x_q \quad x''_q \quad T''_{qo}]^T$ is the vector containing the synchronous generator parameters, $u = [\Delta V_q \Delta \quad \Delta V_d \quad V_{fd}]^T$, represents the input, $x(t) = [\Delta E'_q \quad \Delta E''_q \quad \Delta E''_d]^T$, are the state variables, $y(t) = [\Delta i_d \quad \Delta i_q]^T$ is the output.

Then the Jaccobian is defined by:

$$\mathbf{J} = \left. \frac{\partial \hat{\mathbf{y}}_i(p)}{\partial p_r} \right|_{p=\hat{\mathbf{p}}} \qquad (A.6)$$

where each row is given by

$$\mathbf{J}_i = \frac{\partial \mathrm{r}_i}{\partial \mathbf{p}} = -\frac{\partial \hat{\mathrm{y}}_i}{\partial \mathbf{p}} \qquad (A.7)$$

and is computed by integrating the sensitivity equation

$$\begin{aligned} \frac{\partial \dot{\mathbf{x}}(\mathbf{t},\mathbf{p})}{\partial \hat{p}_i} &= \mathbf{A}(\mathbf{p})\frac{\partial \mathbf{x}(\mathbf{t})}{\partial \hat{p}_i} + \frac{\partial \mathbf{A}(\mathbf{p})}{\partial \hat{\alpha}_i}\mathbf{x}(\mathbf{t}) + \frac{\partial \mathbf{B}(\mathbf{p})}{\partial \hat{p}_i}\mathbf{u}(\mathbf{t}) \\ \frac{\partial \mathbf{y}(\mathbf{t},\boldsymbol{\alpha})}{\partial \hat{p}_i} &= \mathbf{C}(\mathbf{p})\frac{\partial \mathbf{x}(\mathbf{t})}{\partial \hat{p}_i} + \frac{\partial \mathbf{C}(\mathbf{p})}{\partial \hat{p}_i}\mathbf{x}(\mathbf{t}) + \frac{\partial \mathbf{D}(\mathbf{p})}{\partial \hat{p}_i}\mathbf{u}(\mathbf{t}) \end{aligned} \qquad (A.8)$$

Notice that the last equation can constitute another system of equations, if we assume that $\mathbf{S}_i = \dfrac{\partial \mathbf{x}}{\partial \hat{\alpha}_i}$ (Khalil, 1996). Then is possible to rewrite (A.3) as;

$$\begin{aligned}\dot{\mathbf{S}}_i &= \mathbf{A}(\mathbf{p})\mathbf{S}_i + \frac{\partial \mathbf{A}(\mathbf{p})}{\partial \hat{p}_i}\mathbf{x}(\mathbf{t}) + \frac{\partial \mathbf{B}(\mathbf{p})}{\partial \hat{p}_i}\mathbf{u}(\mathbf{t}) \\ \frac{\partial \mathbf{y}(\mathbf{t})}{\partial \hat{p}_i} &= \mathbf{C}(\mathbf{p})\mathbf{S}_i + \frac{\partial \mathbf{C}(\mathbf{p})}{\partial \hat{p}_i}\mathbf{x}(\mathbf{t}) + \frac{\partial \mathbf{D}(\mathbf{p})}{\partial \hat{p}_i}\mathbf{u}(\mathbf{t})\end{aligned} \qquad (A.9)$$

or equivalently in state space form

$$\begin{aligned}\dot{\mathbf{S}}_i &= \mathbf{A}(\mathbf{p})\mathbf{x} + \begin{bmatrix} \frac{\partial \mathbf{A}(\mathbf{p})}{\partial \hat{p}_i} & \frac{\partial \mathbf{B}(\mathbf{p})}{\partial \hat{p}_i} \end{bmatrix} \begin{bmatrix} \mathbf{x}(\mathbf{t}) \\ \mathbf{u}(\mathbf{t}) \end{bmatrix} \\ \frac{\partial \mathbf{y}(\mathbf{t})}{\partial \hat{p}} &= \mathbf{C}(\mathbf{p})\mathbf{x} + \begin{bmatrix} \frac{\partial \mathbf{C}(\boldsymbol{\alpha})}{\partial \hat{p}_i} & \frac{\partial \mathbf{D}(\mathbf{p})}{\partial \hat{p}_i} \end{bmatrix} \begin{bmatrix} \mathbf{x}(\mathbf{t}) \\ \mathbf{u}(\mathbf{t}) \end{bmatrix}\end{aligned} \qquad (A.10)$$

and

$$\mathbf{J} = -\begin{bmatrix} \frac{\partial \hat{\mathbf{y}}(t_i)}{\partial \mathbf{p}} \\ \vdots \\ \frac{\partial \hat{\mathbf{y}}(t_N)}{\partial \mathbf{p}} \end{bmatrix} \qquad (A.11)$$

A.2 Kalman Matrix A_k C_k

The EKF algorithm requires the calculation of two matrices A_k and C_k, were they are defined by:

$$A_k = \frac{\partial f^e(x_k^e, u_k)}{\partial x_k^{eT} \mid \hat{x}_{k|k}^e} \qquad (A.12)$$

$$C_k = \frac{\partial h^e(x_k^e)}{\partial x_k^{eT} \mid \hat{x}_{k|k-1}^e} \qquad (A.13)$$

For the linear system defined by (A.1) the calculated matrices A_k and C_k are equal to:

$$A_k = \begin{bmatrix} 1+\frac{Ts}{T'_{d0}} & -Ts\frac{(x_d - x'_d)}{(T'_{d0}\, x''_d)} & \\ Ts\left(\frac{1}{T''_{do}} - \frac{1}{T'_{do}}\right) & 1+Ts\left(\frac{1}{T''_{do}} + \frac{xd - x'd}{T'_{do}x''d} + \frac{x'd - x''d}{T''_{do}x''d}\right) & J1^k_{3x9} \\ & 1+\frac{Ts * xq}{T''_{qo}x''_q} & \\ 0_{9x3} & & I_{9x9} \end{bmatrix} \qquad \text{(A.14)}$$

$$C_k' = \begin{bmatrix} 0 & 0 \\ \frac{1}{x_d''} & 0 \\ 0 & \frac{1}{x_q''} \\ 0 & 0 \\ 0 & 0 \\ \frac{V_q}{x_d''} - \frac{E_q''}{x_d''} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -\frac{V_d}{x_q''} - \frac{E_q''}{x_q''} \\ 0 & 0 \end{bmatrix} \qquad \text{(A.15)}$$

were

$$J1^{k}{}' = Ts \begin{bmatrix} -\frac{V_q - E''_q}{T'_{d0}\, x''_d} & \frac{V_q - E''_q}{T'_{d0}\, x''_d} & 0 \\ -\frac{(V_q - E''_q)}{T'_{d0}\, x''_d} & \left(-\frac{1}{T'_{d0}\, x'_d} + \frac{1}{T''_{d0}\, x'_d}\right)(V_q - E''_q) & 0 \\ -\frac{(x_d - x'_d)}{T'_{d0}\, x''^2_d}(V_q - E''_q) & \left(-\frac{(x_d - x'_d)}{T'_{d0}\, x''^2_d} - \frac{x_d}{T''_{d0}\, x''^2_d}\right)(V_q - E''_q) & 0 \\ -\frac{(x_d - x'_d)}{T'^2_{d0}\, x''_d}(V_q - E''_q) - \left(\frac{K-1}{T'^2_{d0}}\right)V_{fd} & -\frac{(x_d - x'_d)(V_q - E''_q)}{T'^2_{d0}\, x''^2_d} - \frac{KV_q}{T'^2_{d0}\, x''_d} & 0 \\ 0 & -\frac{(x_d - x'_d)(V_q - E''_q)}{T''^2_{d0}\, x''^2_d} - \frac{(E'_q - E''_q)}{T''^2_{d0}} & 0 \\ \frac{V_{fd}}{T'_{do}} & \frac{V_{fd}}{T''^2_{d0}} & 0 \\ 0 & 0 & -\frac{(V_d + E''_q)}{T'_{q0}\, x''_q} \\ 0 & 0 & \frac{x_q(V_d - E'_q)}{T''_{q0}\, x''^2_q} \\ 0 & 0 & \frac{(x_q - x''_q)(V_d - E'_q)}{T'^2_{q0}\, x''^2_q} \end{bmatrix} \quad (A.16)$$

The [revios equation were taken for a discrete version of the equation (A.1) for this reason some components of the matrix A_k are multiplied for the sample time Ts.

APPENDIX B
SQUARE ROOT FILTER-DECOUPLED SYSTEM

The analysis of chapters 4 and 5 were focused to present the Extended Kalman Filter and its modifications for parameter identification. However when the calculus is made with finite word length the algorithm usually is mathematically unstable and inaccurate. This section present a square root filter called "UD Covariance Factorization Filter" which modifies the equations of EKF in order to propagate and predict the Covariance Matrix. The UD factorization avoids the matrix inversion and increases the computational burden of the EKF.

As presented in [Maybeck,1979] the numerical instability of Kalman Filter can be reduced using Square Root Filters. The UD Covariance Factorization Filter shows to be the best solution compared with others Square Root Filters like: "Inverse Covariance Square Root", "Inverse covariance" and others. The square root filter decomposes the covariance matrix in the multiplication of three matrixes as presented below:

.

$$P(t) = U(t)D(t)U^T(t) \qquad (B.1)$$

where U is an upper triangular unitary matrix (ones in the principal diagonal) and D is a diagonal matrix. The matrixes U and D are used to propagate and correct the extended state and the covariance matrix avoiding the matrix inversion matrix as the required by equation (4.44). In [Maybeck,1979] is presented how the method was developed, but here is only presented the general algorithm. The Figure 44 presents the algorithm used to decompose the covariance matrix in the U and D matrixes.

The EKF algorithm implementation with the UD factorization requires the implementation of the equations (4.44) and (4.45) by the corrector algorithm presented in Figure 45. This algorithm takes the weighted matrix of the output error, the covariance matrix and the matrix A (equation 4.37) to correct the covariance matrix and calculate the Kalman gain.

Additional to previous modifications, the covariance matrix of the equation (4.43) must be predicted. The Figure 46 implements this equation.

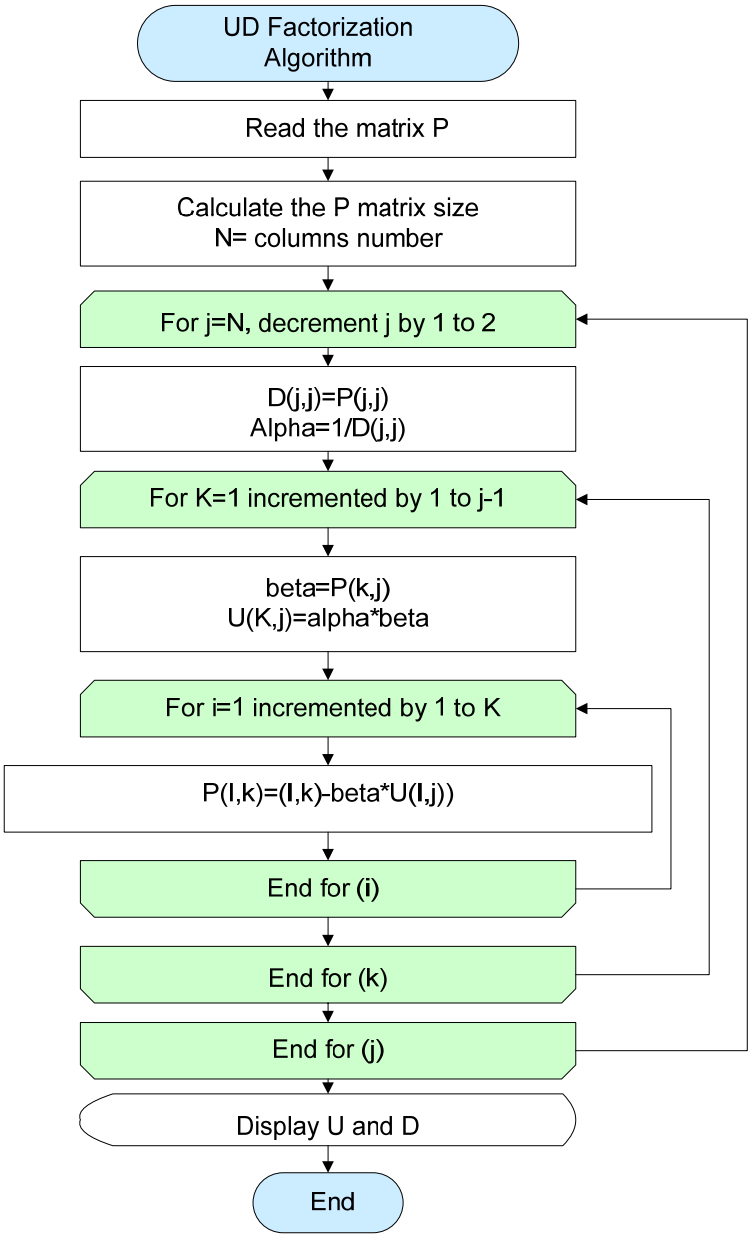

Figure 44 UD factorization algorithm

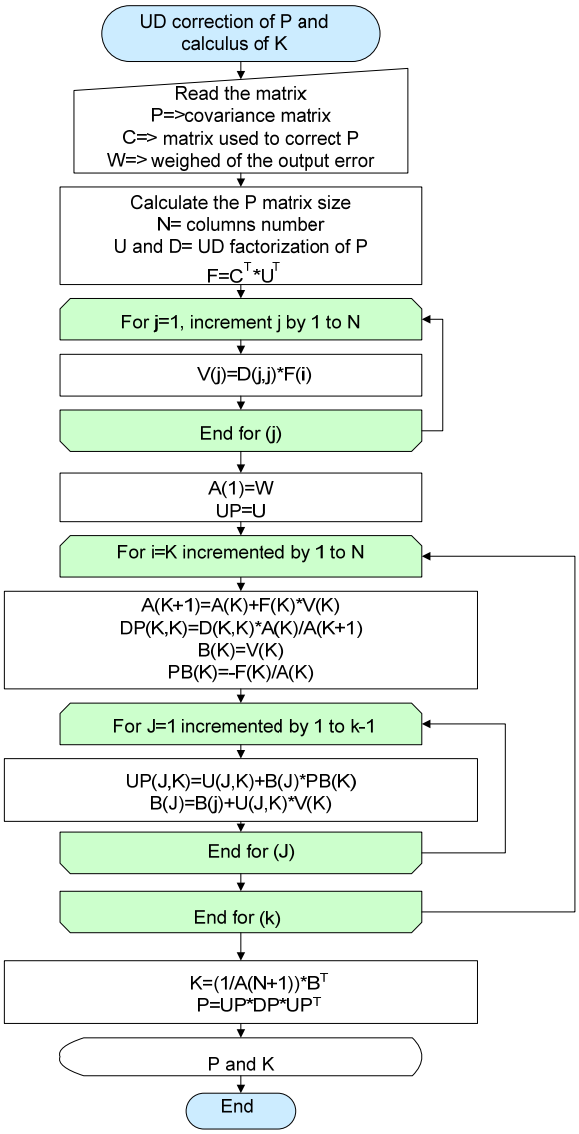

Figure 45 UD factorization (P correction and K calculus)

The application of the EKF with UD factorization requires that the output error must be a scalar value. However for the model presented in section 2 this is a vector of 2 rows by one column. To apply the UD factorization was used and additional modification that decoupled the system in two systems. The simulation results of the chapters 4 and 5 shows a decoupled between the elements in the D axis with the element in the Q axis. This simplification was used with the UD factorization to

generate the final algorithm. The decomposition made two sub systems, the first is composed by the three inputs (V_d, V_{fd} and V_q), the two first states (Eq' and Eq"), Id and the first six parameters (X_d, X_d', X_d", T_{d0}',T_{d0}" and K). The second subsystem is composed by the three inputs (V_d, V_{fd} and V_q), the third state (Ed"), Iq and the last three parameters (X_q, , X_q" and T_{q0}"). The assumption made assume that there is not coupling between these two subsystems.

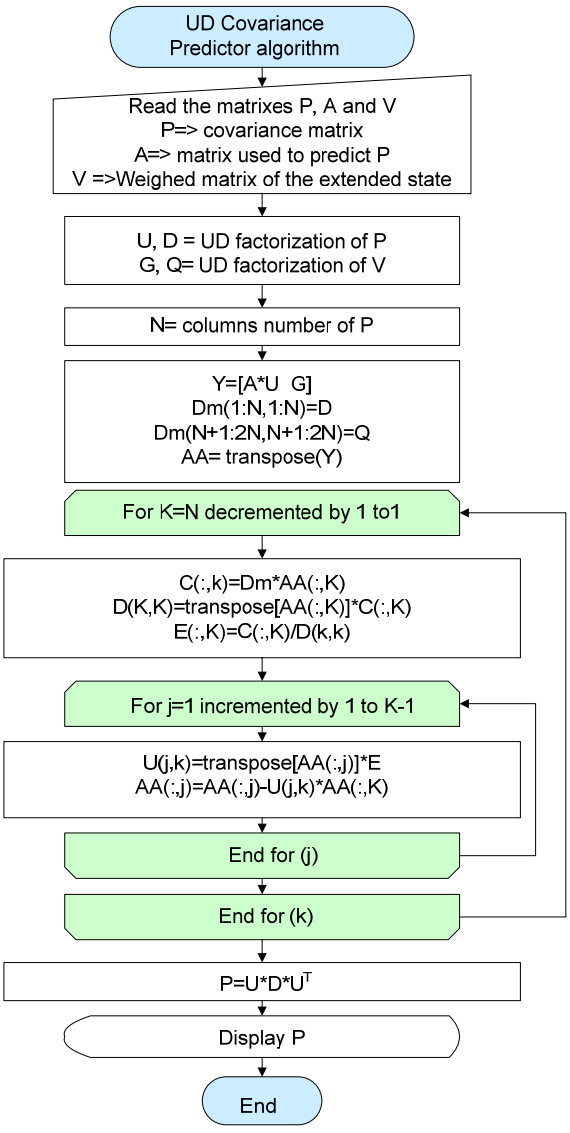

Figure 46 UD factorization (P predictor)

The comparative analysis between the EKF and the UD-EKF presented in Table 9 shows the similar performance of these two methodologies. Based on the simulations, the UD factorization is recommended when the EKF filter is applied in software that don't have the same performance and algorithms to deal with the matrix inversion like MATLAB, that is the case of it application in processors that work in real time.

Table 9 comparative analysis of the EKF and the UD-EKF

Comparative analysis between the EKF with the UD factorized version							
Para-meter	***V/ Real***	***SS-EKF noisy***			***SS-EKF-UD factorization noisy***		
		V/ initial	***V/ esti-mated***	***error %***	***V/ initial***	***V/ esti-mated***	***error %***
Xd	1.414	1.272	1.419	0.45	1.272	1.419	0.36
Xd'	0.333	0.299	0.335	0.86	0.299	0.336	0.82
Xd''	0.208	-	-	-	-	-	-
Td0'	5.85	5.265	5.820	0.50	5.265	5.816	0.57
Td0''	0.194	0.174	0.196	1.02	0.174	0.196	1.05
K	1552	-	-	-	-	-	-
Xq	1.302	1.171	1.302	0.02	1.171	1.302	0.00
Xq''	0.396	0.356	0.485	3.19	0.356	0.405	2.29
Tq0''	0.955	0.859	0.954	0.10	0.859	0.946	0.96

APPENDIX C
MATLAB CODE FILES FOR EKF

This appendix shows the MATLAB® codes and files for computing the parameter the Extended Kalman filer applied to synchronous generator problem. The files required to generate the impost-outputs to the systems can be found in [Jauregui, 2001]. The table presented below summarizes the code files presented in this algorithm.

Function	Description
EKF_CODE	This algorithm apply the general structure to calculate the EKF
esta0.m	This file calculates recursively the states of the system
obs_kal.m	Calculates the propagation and prediction equations of the EKF
jac_rls.m	This file calculates recursively the Jacobian of the system
graficar.m	File used to plot the different waveforms used to analyze the EKF

Table 10 EKF code files

C.1 General algorithm of EKF

```
% this program Calculates the parameters of a synchronous generaror
% by the Extended Kalman Filter algorithm
clc
clear all
close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%              step 1              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   charge the imput-outputs of the systems   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
format long
load vfd.mat
load T.mat
load vd.mat
load vq.mat
load ye.mat
load Eq_.mat
load Eq__.mat
load Ed__.mat
id=ye(:,1);
iq=ye(:,2);
```

```
datoa0=[1.414 0.333 0.208 5.85 0.194 1552 1.302 0.396 0.955]';
P1=datoa0;
% data0(1)=xd          % 1. parameter to be estimated
% data0(2)=xd_              % 2. parameter to be estimated
% data0(3)=xd__          % 3. parameter to be estimated
% data0(4)=Td0_          % 4. parameter to be estimated
% data0(5)=Td0__         % 5. parameter to be estimated
% data0(6)=k             % 6. parameter to be estimated
% data0(7)=xq        % 7. parameter to be estimated
% data0(8)=xq__          % 8. parameter to be estimated
% data0(9)=Tq0__         % 9. parameter to be estimated

T0=1;
M=1024;
N=M;
A=eye(5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%              step  2              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mode Selection: Setup the estimation method %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mode_1=2;

%modo 1 full estimation           EKF
%modo 2 full estimation noise       EKF_N



    indiG=[1 2 3 4 5 6 7 8 9 10 11 12 ];
    etr=0.9;
    P0=etr*[1.414 0.333 0.208 5.85 0.194 1552 1.302 0.396 0.955]';
    NNN=1;
    W(1,1)=50;
    W(2,2)=50000;
    Vx=0.1*eye(3);
    Vp=2000*eye(9);
    Vp(1,1)=4000;
    Vp(3,3)=3000;
    Vp(5,5)=2500;
    Vp(2,2)=1000;
    PKE=1000*eye(12);

    Vp(7,7)=10000;
    Vp(8,8)=2500;
    Vp(9,9)=100000;

    PKE(3,3)=1e10;
    PKE(10,10)=1e10;
    PKE(11,11)=1e10;
```

```
    PKE(12,12)=1e10;

% V     ==> eigth matrix of the Variables estimated
% P_C   ==> Matrix added to compensate the selection
% X0    ==> initialization of the variable states
% TS    ==> sampling time
% p     ==> initial value  of estimated parameter
% ppa   ==> matrix uset to plot the parameter estimation througth the time
V(1:3,1:3)=Vx;
V(4:12,4:12)=Vp;
P_C=100*eye(12-length(indiG));
X0=[0; 0; 0];
TS=T(2)-T(1);
p=P0;
ppa=zeros(N,9);

% xT    ==>
% xT_1  ==>
% dxT_1 ==>
%%%%%%%%%%%%%%%%%
xT=[0;0;0];
xT_1=[0;0;0];
dxT_1=zeros(27,1);
%%%%%%%%%%%%%%%%%

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %%               step  3               %%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Increase of the length of the inputs and    %
% %      outputs to get a size of 2048    optional to see the convergence with more samples     %
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% for i=M:N
%     T(i)=T(i-1)+T(2);
%     vfd(i)=vfd(M);
%     vd(i)=vd(M);
%     vq(i)=vq(M);
%     ye(i,:)=ye(M,:);
%     Eq_(i)=Eq_(M);
%     Eq__(i)=Eq__(M);
%     Ed__(i)=Ed__(M);
% end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%               step  4               %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Noise added to the measured signals      %
%           inuts      outputs              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
switch mode_1
   case{1}
   case{2}
      nn1=randn(N,1);
      nn2=nn1/max([max(nn1) abs(min(nn1))]);
      nn3=randn(N,2);
      nn4=nn3/max([max(max(nn3)) abs(min(min(nn3)))]);
      vd=vd+0.0004*nn2';         %Noise to be added vd voltage vector
      vq=vq+0.0004*nn2';         %Noise to be added vq voltage vector
      vfd=vfd+0.000004*nn2;      %Noise to be added vfd voltage vector
      ye=ye+0.001*nn4;           %Noise to be added ye voltage vector
   otherwise
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                step  5                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        Core of EKF-Subset -iterated        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ppa(1,:)=p';
for i=(T0+1):N

  u=[vd(i-1) vd(i); vq(i-1) vq(i); vfd(i-1) vfd(i)];
  Ti=[0; TS];
  % esta0 Calculate: state==>X0, output==>YY,  error==>e
  [e,X0,YY] = esta0(p,u,Ti,ye(i,:),X0);
  % NNN ==> number of iterations
  % if the number of iteration is greather than 1 is necesary to have
  % the  actual value  and previous value of the states and
  % the  actual value  and previous value of the parameters
  % Xi  ==> previous iteration -variable state value
  % Xi  ==> previous iteration -parameter value

     X1=X0;
     pi=p;
     %obs_kal  ==> function used to implement the extended kalman filter
     % the impust were defined as  comented before
     % the outputs are
     % con p0 ==>conditioning value of covariance matrix
     % con p  ==>conditioning value of covariance matrix forward prediction
     % X0     ==> final variables state estimated
     % p      ==> final parameters estimated
     % PKE    ==> covariance matrix
     % KK     ==> Kalman Gain
     % e      ==> final Error betuen the real output and the estimated
     [con_p0,EV_P,X0,p,PKE,KK,e] = obs_kal(p,ye(i,:),X0,PKE,W,V,u,TS,indiG);
  % errrr   ==>Time evolution matrix used to plot the error
  % ppa     ==>Time evolution matrix used to plot the estimated parameters
  % XA      ==>Time evolution matrix used to plot the estimated variables states
  % YA      ==>Time evolution matrix used to plot the estimated outputs
```

```
  % Kd      ==>Time evolution matrix used to plot d axe gains
  % Kq      ==>Time evolution matrix used to plot q axis
  % con_p0f ==>Time evolution matrix used to plot covariance matrix
  % con_pf  ==>Time evolution matrix used to plot covariance matrix forward prediction
  errrr(:,i)=e;
  ppa(i,:)=p';
  XA(:,i)=X0;
  YA(:,i)=YY;
  Kd(:,i)=KK(:,1);
  Kq(:,i)=KK(:,2);
  %%%%%%%%%%%%%%%%%%%%%%%
  con_p0f(i)=con_p0;
  EV_PF(i,:)=sort(EV_P);
  %%%%%%%%%%%%%%%%%%%%%%%%

  %^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
  %%                 step  5x               %%
  %^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
  %             Jacobian calculus             %
  %^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
  % XXa   ==> variable state value
  % uT    ==> imput in the actual time
  % uT    ==> imput in the previous time
  % xT    ==> variable state value in the actual time
  % xT_1  ==> variable state value in the previous time
  XXa(:,i)=X0;
  uT=[vd(i) ;vq(i); vfd(i)];
  uT_1=[vd(i-1) ;vq(i-1); vfd(i-1)];
  xT(:,1)=XXa(:,i);
  xT_1(:,1)=XXa(:,i-1);
  %%%%%%%%%%%%%%%%%%%%%%%
  % jac_rls ==> fuction used to calculate the jacobian in recursive way
  % the impust were defined as  comented before
  % the outputs are:
  % dyT ==> jacobian of the output respect to the estimated parameteres
  % dyT ==> jacobian of the variables states respect to the estimated parameteres
  [dyT,dxT_1] = jac_rls(p,uT,uT_1,xT,xT_1,dxT_1);
  % jacobiano  ==>Time evolution matrix used to plot jacobian matrix
  jacobiano(:,(2*i+1):(2*i+2))=-dyT(:,1:2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                 step  6               %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Getting final results          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% H    ==> Hessian matrix
% ei_v ==> Eigenvalues of H
% P    ==> Final parameter errors
fprintf(' Hessian Matrix \n')
```

```
H=jacobiano*jacobiano'
fprintf(' Eigenvalues of H \n')
ei_v=eig(H)
fprintf(' Final parameter errors in %       \n')
for i=1:9
   PE(i)=(P1(i)-p(i))*100/P1(i);
   PEB(i)=(PE(i));
end
PE';
param=['xd       ' ; 'xd`      ' ; 'xd``     '; 'Td0`     ';  'Td0``    ' ; 'k        ' ; 'xq       ';  'xq``     ' ; 'Tq0``    '];
valo=['Parameter ' '       Real Value' '    Initial Value' '  Estimated value' '         % error'];
mm1=[ P1 P0 p PEB'];
RES_1=valo;
RES_1(2:10,1:10)=param;
for i=1:4
   dat_1=num2str(mm1(:,i));
   [ty tx]=size(dat_1);
   RES_1(2:ty+1,(10+17*i-tx+1):(10+17*i))=dat_1;
end
RES_1



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                 step  7                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Results plots              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Graficar
```

C.2 Recursive State propagation algorithm (“esta0.m”)

```
% This algorithm Propagates the state in recursive way
%
%^^ inputs ^^^
%X0     ==> initial condition of the state
%y      ==> measured output
%T      ==> time used to propagate the state
%u      ==> input to the linear system
%p      ==> parameter vector used to generate the linear system
%
%^^ outputs ^^^
% YY    ==> propagated output
% X1    ==> propagated state
% e     ==> error betwen the propaated output and the measured output
function [e,X1,YY] = esta0(p,u,T,y,X0)

xd=p(1);% 1. parameter to be estimated
xd_=p(2);           % 2. parameter to be estimated
xd__=p(3);          % 3. parameter to be estimated
```

```
Td0_=p(4);          % 4. parameter to be estimated
Td0__=p(5);         % 5. parameter to be estimated
k=p(6);             % 6. parameter to be estimated
xq=p(7);            % 7. parameter to be estimated
xq__=p(8);          % 8. parameter to be estimated
Tq0__=p(9);         % 9. parameter to be estimated

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Linear system Matrix A %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a11=1/Td0_;
a12=-(xd-xd_)/(Td0_*xd__);
a21=1/Td0__-1/Td0_;
a22=-(1/Td0__+(xd-xd_)/(Td0_*xd__)+(xd_-xd__)/(Td0__*xd__));
a33=-xq/(Tq0__*xq__);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Linear system Matrix B %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
b12=(xd-xd_)/(Td0_*xd__);
b13=k/Td0_;
b22=(xd-xd_)/(Td0_*xd__)+(xd_-xd__)/(Td0__*xd__);
b23=k/Td0_;
b31=-(xq-xq__)/(Tq0__*xq__);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Linear system Matrix C %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c12=1/xd__;
c23=1/xq__;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Linear system Matrix D %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
d12=-1/xd__;
d21=1/xq__;
Ax=[a11 a12 0;a21 a22 0;0 0 a33];
Bx=[0 b12 b13;0 b22 b23;b31 0 0];
Cx=[0 c12 0;0 0 c23];
Dx=[0 d12 0;d21 0 0];
sys1=ss(Ax,Bx,Cx,Dx);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%linear integration to propagate the state %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[yx,T,XX]=lsim(sys1,u,T,X0);
X1=XX(2,:)';
YY=yx(2,:)';
e=y'-YY;
```

C.3 Prediction a correction of the parameters and the status (“obs_kal.m”)

```
% This algorithm correct the state and propagates the kalman gain and the error covariance matrix
%
%^^ Inputs ^^
% p     ==> Parameter vector
% YY    ==> Measured output
% X0    ==> propagated state by linear inetgration (linear system)
% PKE   ==> initial error covariance matrix
% W     ==> weighed sensor matrix (related to YY)
% V     ==> weighed estimated parameter matrix (related to p)
% u1    ==> input to the linear system (voltages)
% Ts    ==> sampling time
% indi2 ==> indices of the parameters to be estimated
%
% ^^ Outputs ^^
% con_p0==> condition number of error covariance matrix P
% EV_P  ==> eigenvalues of error covariance matrix P
% x1    ==> corrected states
% p     ==> estimated parameteres vector
% PKM   ==> propagated and corrected error covariance matrix
% KKK   ==> kalman gain
% e     ==> error betwen the measures and the correted estimated output


function [con_p0,EV_P,x1,p1,PKM,KKK,e] = obs_kal(p,YY,X0,PKE,W,V,u1,Ts,indi2)
l=1;
NNN=1;
pi=p;
Xi=X0;
indiT=[1 2 3 4 5 6 7 8 9 10 11 12];
u=u1(:,2);

xd=p(1);% 1. parameter to be estimated
xd_=p(2);           % 2. parameter to be estimated
xd__=p(3);          % 3. parameter to be estimated
Td0_=p(4);          % 4. parameter to be estimated
Td0__=p(5);         % 5. parameter to be estimated
k=p(6);             % 6. parameter to be estimated
xq=p(7);% 7. parameter to be estimated
xq__=p(8);          % 8. parameter to be estimated
Tq0__=p(9);         % 9. parameter to be estimated

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Propagation of the output and calculus of the error betwen   %
% the measured and the estimated output                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
c12=1/xd__;
c23=1/xq__;
```

```
Cx=[0 c12 0;0 0 c23];
d12=-1/xd__;
d21=1/xq__;
Dx=[0 d12 0;d21 0 0];
e=YY'-(Cx*X0+Dx*u);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% diference H ( output equations) respect to the parameteres     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Dm=zeros(18,6);
Dm(3,2)=1/(xd__^2);
Dm(3,5)=-1/(xd__^2);
Dm(17,1)=-1/(xq__^2);
Dm(17,6)=-1/(xq__^2);

CPK=[Dm(1:9,1:6)*[u;X0] Dm(10:18,1:6)*[u;X0] ];
CK=zeros(2,12);
CK(1:2,1:3)=Cx;
CK(1:2,4:12)=CPK';

% Ck ==> C matrix of the extented kalman filter related to the output
% equation
PD1=setdiff(indiT,indi2);

% KK    ==>calculus of Kalman Gain
   KK=PKE*CK'*inv(W+CK*PKE*CK');
% PKE0  ==> correction of error covariance matrix
   PKE0=PKE-KK*CK*PKE;
% Xk    ==>extended state
   XK=[X0; p];
   XKi=[Xi; pi];
% XK    ==> corrention of the extended state
   XK=XK+KK*(e);
% calculus of the contition number of the covariance matrix
   con_p0=cond(PKE0);
% calculus of the eigenvalues of the covariance matrix
   EV_P=eig(PKE0);
% update the states and the parameteres
x1(1:3,1)=XK(1:3,1);
p1(1:9,1)=XK(4:12,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% difference F ( state propagation equations) respect to the parameters     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xd=p1(1);       % 1. parameter to be estimated
xd_=p1(2);      % 2. parameter to be estimated
xd__=p1(3);     % 3. parameter to be estimated
Td0_=p1(4);     % 4. parameter to be estimated
Td0__=p1(5);      % 5. parameter to be estimated
k=p1(6);            % 6. parameter to be estimated
xq=p1(7);       % 7. parameter to be estimated
xq__=p1(8);     % 8. parameter to be estimated
Tq0__=p1(9);      % 9. parameter to be estimated
```

```
B11=[0 1/Td0_/xd__ 0 0 -1/Td0_/xd__ 0
0 -1/Td0_/xd__ 0 0 1/Td0_/xd__ 0
0 -(xd-xd_)/(Td0_*(xd__^2)) 0 0 (xd-xd_)/(Td0_*(xd__^2)) 0
0 -(xd-xd_)/(xd__*(Td0_^2)) -k/(Td0_^2) -1/(Td0_^2) ...
(xd-xd_)/(xd__*(Td0_^2)) 0
0 0 0 0 0 0
0 0 1/Td0_ 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0];

B21=[0 1/Td0_/xd__ 0 0 -1/Td0_/xd__ 0
0 -1/Td0_/xd__+1/Td0__/xd__ 0 0 1/Td0_/xd__-1/Td0__/xd__ 0
0 -(xd-xd_)/(Td0_*(xd__^2))-xd_/(Td0__*(xd__^2)) 0 0 ...
(xd-xd_)/(Td0_*(xd__^2))+xd_/(Td0__*(xd__^2)) 0
0 -(xd-xd_)/((Td0_^2)*xd__) -k/(Td0_^2) 1/(Td0_^2) ...
(xd-xd_)/((Td0_^2)*xd__) 0
0 -(xd_-xd__)/((Td0__^2)*xd__) 0 -1/(Td0__^2) ...
(xd_-xd__)/((Td0__^2)*xd__)+1/(Td0__^2) 0
0 0 1/Td0_ 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0];
B31=[zeros(6)
-1/Tq0__/xq__ 0 0 0 0 -1/Tq0__/xq__
xq/(Tq0__*(xq__^2)) 0 0 0 0 xq/(Tq0__*(xq__^2))
(xq-xq__)/((Tq0__^2)*xq__) 0 0 0 0 xq/((Tq0__^2)*xq__)];
% aproximation of the matrix A of EKF by discrete system (Ts)
APK=Ts*[B11*[u;x1] B21*[u;x1] B31*[u;x1]];
a11=1/Td0_;
a12=-(xd-xd_)/(Td0_*xd__);
a21=1/Td0__-1/Td0_;
a22=-(1/Td0__+(xd-xd_)/(Td0_*xd__)+(xd_-xd__)/(Td0__*xd__));
a33=-xq/(Tq0__*xq__);
Ax=Ts*[a11 a12 0;a21 a22 0;0 0 a33];
AK2=zeros(12,12);
AK2(1:3,1:3)=Ax+eye(3);
AK2(4:12,4:12)=eye(9);
AK2(1:3,4:12)=APK';
% Prediction of the error covariance matrix
PKE0=AK2*PKE0*AK2'+V;
KKK=KK;
PKM=PKE0;
```

C.4 Recursive Jacobian calculation (“jac_rls.m”)

```
% function used to calculate recursively the jacobian for
% sensitivity analysis
%
% ^^ inputs ^^
```

```
% dxT_1 ==> initial jacobian of the output respect to the estimated parameters
% uT    ==> input in the actual time
% uT_1  ==> input in the previous time
% xT    ==> variable state value in the actual time
% xT_1  ==> variable state value in the previous time
% p     ==> parameteres vector
%
% ^^ outputs ^^
% dxT   ==> jacobian of the states respect to the estimated parameters
% dyT   ==> jacobian of the output respect to the estimated parameters
function [dyT,dxT] = jac_rls(p,uT,uT_1,xT,xT_1,dxT_1)

% sampling time
Ts=0.01171875000000;
xd=p(1);% 1. parameter to be estimated
xd_=p(2);         % 2. parameter to be estimated
xd__=p(3);        % 3. parameter to be estimated
Td0_=p(4);        % 4. parameter to be estimated
Td0__=p(5);       % 5. parameter to be estimated
k=p(6);           % 6. parameter to be estimated
xq=p(7);% 7. parameter to be estimated
xq__=p(8);        % 8. parameter to be estimated
Tq0__=p(9);       % 9. parameter to be estimated


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% difference of f (state equation) respect to the state variables %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a11=1/Td0_;
a12=-(xd-xd_)/(Td0_*xd__);
a21=1/Td0__-1/Td0_;
a22=-(1/Td0__+(xd-xd_)/(Td0_*xd__)+(xd_-xd__)/(Td0__*xd__));
a33=-xq/(Tq0__*xq__);

A=Ts*[a11 a12 0;a21 a22 0;0 0 a33];
AK=A+eye(3);
for i=1:3
   for j=1:3
      Ax((9*i-8):9*i,(9*j-8):9*j)=AK(i,j)*eye(9);
   end
end

c12=1/xd__;
c23=1/xq__;
Cx((1*9-8):1*9,(2*9-8):2*9)=c12*eye(9);
Cx((2*9-8):2*9,(3*9-8):3*9)=c23*eye(9);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%
% difference of f (state equation) respect to the estimated parameteres %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%

B11=[0 1/Td0_/xd__ 0 0 -1/Td0_/xd__ 0
0 -1/Td0_/xd__ 0 0 1/Td0_/xd__ 0
0 -(xd-xd_)/(Td0_*(xd__^2)) 0 0 (xd-xd_)/(Td0_*(xd__^2)) 0
0 -(xd-xd_)/(xd__*(Td0_^2)) -k/(Td0_^2) -1/(Td0_^2) ...
(xd-xd_)/(xd__*(Td0_^2)) 0
0 0 0 0 0 0
0 0 1/Td0_ 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0];

B21=[0 1/Td0_/xd__ 0 0 -1/Td0_/xd__ 0
0 -1/Td0_/xd__+1/Td0__/xd__ 0 0 1/Td0_/xd__-1/Td0__/xd__ 0
0 -(xd-xd_)/(Td0_*(xd__^2))-xd_/(Td0__*(xd__^2)) 0 0 ...
(xd-xd_)/(Td0_*(xd__^2))+xd_/(Td0__*(xd__^2)) 0
0 -(xd-xd_)/((Td0_^2)*xd__) -k/(Td0_^2) 1/(Td0_^2) ...
(xd-xd_)/((Td0_^2)*xd__) 0
0 -(xd_-xd__)/((Td0__^2)*xd__) 0 -1/(Td0__^2) ...
(xd_-xd__)/((Td0__^2)*xd__)+1/(Td0__^2) 0
0 0 1/Td0_ 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0];

B31=[zeros(6)
-1/Tq0__/xq__ 0 0 0 0 -1/Tq0__/xq__
xq/(Tq0__*(xq__^2)) 0 0 0 0 xq/(Tq0__*(xq__^2))
(xq-xq__)/((Tq0__^2)*xq__) 0 0 0 0 xq/((Tq0__^2)*xq__)];

Bx=[B11;B21;B31];
Bx=Bx*Ts;
Dx=zeros(18,6);

Dx(3,2)=1/(xd__^2);
Dx(3,5)=-1/(xd__^2);
Dx(17,1)=-1/(xq__^2);
Dx(17,6)=-1/(xq__^2);
Dx=Dx;
% Propagation of the jacobian

dxT=Bx*[uT_1;xT_1]+Ax*dxT_1;
dyTa=Dx*[uT;xT]+Cx*dxT;
dyT1=dyTa(1:9);
dyT2=dyTa(10:18);
dyT=[dyT1 dyT2];
```

C.5 Function used to plot the results ("graficar.m")

```
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%              step  7.1             %%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%              imput plot              %
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%

figure(1)
set(gcf, 'color', 'white');
plot(T,[vq;vd;vfd'])
title('System Inputs')
legend('Vq (Quadrature Voltage axis)','Vd (Direct Voltage axis  )','Vfd (Field Voltage)','location','B' )
xlabel('Time [s]')
ylabel('Voltage [V]')


%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%              step  7.2             %%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%           State Variables             %
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%

figure(2)
set(gcf, 'color', 'white');
hold on
plot(T,[Eq_ Eq__ Ed__],':')
plot(T,XA)
title('State Variables')
legend('Eq`  real ','Eq`` real','Ed`` real','Eq`   estimated ','Eq`` estimated','Ed`` estimated','location', 'NE' )
xlabel('Time [s]')
ylabel('Voltage [V]')
hold off


%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%              step  7.3             %%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%              outputs               %
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%

figure(3)
set(gcf, 'color', 'white');
hold on
plot(T,ye,':')
plot(T,YA)
title('System Outputs')
legend('Id real ','Iq real','Id estimated ',...
     'Iq estimated','location', 'B' )
```

```
xlabel('Time [s]')
ylabel('Current [A]')
hold off

%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%              step  7.4              %%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%       parameteres estimated            %
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%

figure(4)
set(gcf, 'color', 'white');

title('Parameters estimated')
xlabel('Time [s]')
ylabel('Value [PU]')
for i=1:9
    subplot(5,2,i)
    asd=ppa(:,i);
    hold on
    plot(T,P1(i)*ones(1,N),':')
    plot(T,asd)
    legend(param(i,:),'location', 'B')
    hold off
end
xlabel({'Time [s]';'Parameters estimated'})



%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%              step  7.5              %%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%       Sensitivity analysis             %
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%

figure(5)
set(gcf, 'color', 'white');
xlabel('Time [s]')
ylabel(' Sentivity Value ')
hold on
title('Sensitivity Analisys')
rf=jacobiano;
for i=1:N
   for j=1:9
      pp2(j,i)=rf(j,2*i-1);
      pp1(j,i)=rf(j,2*i);
   end
end

   subplot(5,2,1)
   asd=pp2(1,:);
```

```
    plot(T,-asd)
    legend('d(Id)/d(xd) ','location', 'B')

    subplot(5,2,2)
    asd=pp2(2,:);
    plot(T,-asd)
    legend('d(Id)/d(xd`) ','location', 'B')

    subplot(5,2,3)
    asd=pp2(3,:);
    plot(T,-asd)
    legend('d(Id)/d(xd``) ','location', 'B')

    subplot(5,2,4)
    asd=pp2(4,:);
    plot(T,-asd)
    legend('d(Id)/d(Td0`) ','location', 'B')

    subplot(5,2,5)
    asd=pp2(5,:);
    plot(T,-asd)
    legend('d(Id)/d(Td0``) ','location', 'B')

    subplot(5,2,6)
    asd=pp2(6,:);
    plot(T,-asd)
    legend('d(Id)/d(K) ','location', 'B')

    subplot(5,2,7)
    asd=pp1(7,:);
    plot(T,-asd)
    legend('d(Iq)/d(xq) ','location', 'B')

    subplot(5,2,8)
    asd=pp1(8,:);
    plot(T,-asd)
    legend('d(Iq)/d(xq``) ','location', 'B')

    subplot(5,2,9)
    asd=pp1(9,:);
    plot(T,-asd)
    legend('d(Iq)/d(Tq0``) ','location', 'B')
    xlabel({'Time [s]';'Sensitivity Analisys'})
hold off


%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%            step  7.6            %%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%    conditioning analysis              %
```

```
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%

figure(6)
set(gcf, 'color', 'white');
hold on
title('Conditioning Of Covariace Matrix P')
plot(T,con_p0f)
xlabel('Time [s]')
ylabel('Conditioning Value')
hold off

%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%%              step  7.7              %%
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
%       Eigevakues of P analysis          %
%^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%

figure(7)

set(gcf, 'color', 'white');
hold on
title('Eigenvalues of P')
xlabel('Time [s]')
ylabel('Eigenvalue')
for i=1:length(EV_P)
    subplot(6,2,i)
    plot(T,EV_PF(:,i))
end
xlabel({'Time [s]';'Eigenvalues of P'})
hold off
```

APPENDIX D
MATLAB CODE SUBSET SELECTION-EKF AND ITERATED EKF

This appendix shows the MATLAB® codes and files for computing the parameter the Iterated Extended Kalman filer and the subset selection methodology applied to synchronous generator problem. Additional required files are presented in the appendix C.The table presented below summarizes the code files presented in this algorithm.

Table 11 IEKF and sub set selection code files

Function	Description
IEKF_SS_CODE	This algorithm apply the general structure to calculate the IEKF and the Sub set selection methodology
obs_kal_1.m	Calculates the propagation and prediction equations of the IEKF

D.1 Subset Selection-EKF and Iterated EKF

```
% this program Calculates the parameters of a synchronous generaror
% by the Extended Kalman Filter algorithm
% Capabilities of the algorithm
%
% * traditional EKF
% * subset selection + EKF
% * iterative extended Kalman Filter
% * subset selection + IEKF
clc
clear all
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%              step 1              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   charge the imput-outputs of the systems   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
format long
load vfd.mat
load T.mat
load vd.mat
load vq.mat
load ye.mat
load Eq_.mat
```

```
load Eq__.mat
load Ed__.mat
id=ye(:,1);
iq=ye(:,2);



datoa0=[1.414 0.333 0.208 5.85 0.194 1552 1.302 0.396 0.955]';
P1=datoa0;
% data0(1)=xd            % 1. parameter to be estimated
% data0(2)=xd_                % 2. parameter to be estimated
% data0(3)=xd__             % 3. parameter to be estimated
% data0(4)=Td0_             % 4. parameter to be estimated
% data0(5)=Td0__            % 5. parameter to be estimated
% data0(6)=k                % 6. parameter to be estimated
% data0(7)=xq           % 7. parameter to be estimated
% data0(8)=xq__             % 8. parameter to be estimated
% data0(9)=Tq0__            % 9. parameter to be estimated

T0=1;
M=1024;
N=M;
A=eye(5);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                step  2               %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Mode Selection: Setup the estimation method %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mode_1=12;

%modo 1  full estimation                EKF
%modo 2  full estimation noise            EKF_N
%modo 3  best subset selection            EKF_SS
%modo 4  -----subset selection noise       EKF_SS_N
%modo 5  -----subset selection            EKF_SS
%modo 6  best subset selection noise       EKF_SS_N
%modo 7  sug by lida ss a               EKF_SS
%modo 8  sug by lida ss b               EKF_SSS
%modo 9  best ss-it-ekf                EKF-SS-IT
%modo 10 ---- ss-it-ekf  noise            EKF-SS-IT_N
%modo 11 ---- ss-it-ekf                EKF-SS-IT
%modo 12 best ss-it-ekf  noise            EKF-SS-IT_N


switch mode_1
   %
   % indiG ==>  variables estimated:  1-3  variable states 4-12 parameteres
   % P0    ==> initial values
   % W     ==> eigth matrix of the outputs
```

```
% Vx    ==> eigth matrix of the state variables estimated
% Vp    ==> eigth matrix of the parameteres estimated
% PKE   ==> initial value of covariance matrix
% NNN   ==> number of iterations
case {1,2}

   indiG=[1 2 3 4 5 6 7 8 9 10 11 12 ];
   etr=0.9;
   P0=etr*[1.414 0.333 0.208 5.85 0.194 1552 1.302 0.396 0.955]';
   NNN=1;
   W(1,1)=50;
   W(2,2)=50000;
   Vx=0.1*eye(3);
   Vp=2000*eye(9);
   Vp(1,1)=4000;
   Vp(3,3)=3000;
   Vp(5,5)=2500;
   Vp(2,2)=1000;
   PKE=100000*eye(12);

   Vp(7,7)=10000;
   Vp(8,8)=2500;
   Vp(9,9)=100000;

   PKE(3,3)=1e10;
   PKE(10,10)=1e10;
   PKE(11,11)=1e10;
   PKE(12,12)=1e10;
   if mode_1==1
      noise_1=0;
   else
      noise_1=1;
   end

case {3,4}

   indiG=[1 2 3 4 5 6 7  8  10 11 12];
   etr=0.9;
   P0=etr*[1.414 0.333 0.208 5.85 0.194 1552/etr 1.302 0.396 0.955]';
   NNN=1;
   W(1,1)=700000;
   W(2,2)=200000;
   Vx=0.1*eye(3);
   Vp=2000*eye(9);
   Vp(1,1)=7000;
   Vp(2,2)=1000;
   Vp(3,3)=5500;
   Vp(4,4)=150000;
   Vp(5,5)=2500;
   Vp(6,6)=1000;
   PKE=100000*eye(12);
```

```
    Vp(7,7)=10000;
    Vp(8,8)=2500;
    Vp(9,9)=100000;

    PKE(3,3)=1e10;
    PKE(10,10)=1e10;
    PKE(11,11)=1e10;
    PKE(12,12)=1e10;
    if mode_1==3
       noise_1=0;
    else
       noise_1=1;
    end

case {5,6}

    indiG=[1 2 3 4 5 7  8  10 11 12 ];
    etr=0.9;
    P0=etr*[1.414 0.333 0.208/etr 5.85 0.194 1552/etr 1.302 0.396 0.955]';
    NNN=1;
    W(1,1)=700000;
    W(2,2)=200000;
    Vx=0.1*eye(3);
    Vp=2000*eye(9);
    Vp(1,1)=7000;
    Vp(2,2)=1000;
    Vp(3,3)=6000;
    Vp(4,4)=160000;
    Vp(5,5)=2400;
    Vp(6,6)=100;
    Vp(8,8)=1900;
    Vp(9,9)=100;
    PKE=100000*eye(12);

    Vp(7,7)=10000;
    Vp(8,8)=2500;
    Vp(9,9)=100000;

    PKE(3,3)=1e10;
    PKE(10,10)=1e10;
    PKE(11,11)=1e10;
    PKE(12,12)=1e10;
    if mode_1==5
       noise_1=0;
    else
       noise_1=1;
    end

case {7,8}
    if mode_1==7
```

```
        indiG=[1 2 3  5 6 7 8  10 11 12 ];
        etr=0.9;
        P0=etr*[1.414/etr 0.333 0.208 5.85 0.194 1552/etr 1.302 0.396 0.955]';
    else
        indiG=[1 2 3 4 5 6   8  10 11 12];
        etr=0.9;
        P0=etr*[1.414 0.333 0.208 5.85/etr 0.194 1552/etr 1.302 0.396 0.955]';
    end
    NNN=1;
    W(1,1)=700000;
    W(2,2)=200000;
    Vx=0.1*eye(3);
    Vp=2000*eye(9);
    Vp(1,1)=7000;
    Vp(2,2)=1000;
    Vp(3,3)=6000;
    Vp(4,4)=160000;
    Vp(5,5)=2400;
    Vp(6,6)=100;
    Vp(8,8)=1900;
    Vp(9,9)=1000;
    PKE=100000*eye(12);

    Vp(7,7)=10000;
    Vp(8,8)=2500;
    Vp(9,9)=100000;

    PKE(3,3)=1e10;
    PKE(10,10)=1e10;
    PKE(11,11)=1e10;
    PKE(12,12)=1e10;
    noise_1=1;

case {9,10}

    indiG=[1 2 3 4 5 6 7  8 10 11  12];
    etr=0.9;
    P0=etr*[1.414 0.333 0.208 5.85 0.194 1552/etr 1.302 0.396 0.955]';
    NNN=10;
    W(1,1)=180000;
    W(2,2)=700000;
    Vx=0.1*eye(3);
    Vp=2000*eye(9);
    Vp(1,1)=8000;
    Vp(2,2)=600;
    Vp(3,3)=4700;
    Vp(4,4)=150000;
    Vp(5,5)=2300;
    Vp(6,6)=1000;
    Vp(7,7)=6000;
    Vp(8,8)=6000;
```

```
    Vp(9,9)=500000;
    PKE=100000*eye(12);

    Vp(7,7)=10000;
    Vp(8,8)=2500;
    Vp(9,9)=100000;

    PKE(3,3)=1e10;
    PKE(10,10)=1e10;
    PKE(11,11)=1e10;
    PKE(12,12)=1e10;
    if mode_1==9
       noise_1=0;
    else
       noise_1=1;
    end


  case {11,12}

    indiG=[1 2 3 4 5 7  8 10 11 12 ];
    etr=0.9;
    P0=etr*[1.414 0.333 0.208/etr 5.85 0.194 1552/etr 1.302 0.396 0.955]';
    NNN=10;
    W(1,1)=180000;
    W(2,2)=700000;
    Vx=0.1*eye(3);
    Vp=2000*eye(9);
    Vp(1,1)=8000;
    Vp(2,2)=400;
    Vp(3,3)=1000;
    Vp(4,4)=150000;
    Vp(5,5)=1800;
    Vp(6,6)=1000;
    PKE=100000*eye(12);

    Vp(7,7)=10000;
    Vp(8,8)=1500;
    Vp(9,9)=100000;

    PKE(3,3)=1e10;
    PKE(10,10)=1e10;
    PKE(11,11)=1e10;
    PKE(12,12)=1e10;
    if mode_1==11
       noise_1=0;
    else
       noise_1=1;
    end
  otherwise
end
```

```
% V     ==> eigth matrix of the Variables estimated
% P_C   ==> Matrix added to compensate the selection
% X0    ==> initialization of the variable states
% TS    ==> sampling time
% p     ==> initial value  of estimated parameter
% ppa   ==> matrix uset to plot the parameter estimation througth the time
V(1:3,1:3)=Vx;
V(4:12,4:12)=Vp;
X0=[0; 0; 0];
TS=T(2)-T(1);
p=P0;
ppa=zeros(N,9);

% xT    ==>
% xT_1  ==>
% dxT_1 ==>
%%%%%%%%%%%%%%%%
xT=[0;0;0];
xT_1=[0;0;0];
dxT_1=zeros(27,1);
%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                 step  3                 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Increase of the length of the imputs and    %
%      outputs to get a size of 2048          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% for i=M:N
%     T(i)=T(i-1)+T(2);
%     vfd(i)=vfd(M);
%     vd(i)=vd(M);
%     vq(i)=vq(M);
%     ye(i,:)=ye(M,:);
%     Eq_(i)=Eq_(M);
%     Eq__(i)=Eq__(M);
%     Ed__(i)=Ed__(M);
% end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                 step  4                 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Noise added to the measured signals     %
%           inuts      outputs                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if noise_1==1
       nn1=randn(N,1);
```

```
    nn2=nn1/max([max(nn1) abs(min(nn1))]);
    nn3=randn(N,2);
    nn4=nn3/max([max(max(nn3)) abs(min(min(nn3)))]);
    vd=vd+0.0004*nn2';          %Noise to be added vd voltage vector
    vq=vq+0.0004*nn2';          %Noise to be added vq voltage vector
    vfd=vfd+0.000004*nn2;       %Noise to be added vfd voltage vector
    ye=ye+0.001*nn4;            %Noise to be added ye voltage vector
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                 step  5                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        Core of EKF-Subset -iterated        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ppa(1,:)=p';
for i=(T0+1):N

  u=[vd(i-1) vd(i); vq(i-1) vq(i); vfd(i-1) vfd(i)];
  Ti=[0; TS];
  % esta0 Calculate: state==>X0, output==>YY,  error==>e
  [e,X0,YY] = esta0(p,u,Ti,ye(i,:),X0);
  % NNN ==> number of iterations
  % if the number of iteration is greather than 1 is necesary to have
  % the  actual value  and previous value of the states and
  % the  actual value  and previous value of the parameters
  % Xi  ==> previous iteration -variable state value
  % Xi  ==> previous iteration -parameter value
  for l=1:NNN
     switch l
        case 1,
           % storing the initial values if it is applied the itereted filter
           X1=X0;
           pi=p;
           PKEi=PKE;
        otherwise
     end
     %obs_kal  ==> function used to implement the extended kalman filter
     % the impust were defined as  comented before
     % the outputs are
     % con p0 ==>conditioning value of covariance matrix
     % con p  ==>conditioning value of covariance matrix forward prediction
     % X0     ==> final variables state estimated
     % p      ==> final parameters estimated
     % PKE    ==> covariance matrix
     % KK     ==> Kalman Gain
     % e      ==> final Error betuen the real output and the estimated
     [con_p0,EV_P,X0,p,PKE,KK,e] = obs_kal_1(p,pi,ye(i,:),X0,X1,PKE,PKEi,W,V,u,TS,indiG,l,NNN);
  end
  % errrr   ==>Time evolution matrix used to plot the error
```

```
  % ppa    ==>Time evolution matrix used to plot the estimated parameters
  % XA     ==>Time evolution matrix used to plot the estimated variables states
  % YA     ==>Time evolution matrix used to plot the estimated outputs
  % Kd     ==>Time evolution matrix used to plot d axe gains
  % Kq     ==>Time evolution matrix used to plot q axis
  % con_p0f ==>Time evolution matrix used to plot covariance matrix
  % con_pf  ==>Time evolution matrix used to plot covariance matrix forward prediction
  errrr(:,i)=e;
  ppa(i,:)=p';
  XA(:,i)=X0;
  YA(:,i)=YY;
  Kd(:,i)=KK(:,1);
  Kq(:,i)=KK(:,2);
  %%%%%%%%%%%%%%%%%%%%%%%%
  con_p0f(i)=con_p0;
  EV_PF(i,:)=sort(EV_P);
  %%%%%%%%%%%%%%%%%%%%%%%%

  %^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
  %%                step  5x              %%
  %^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
  %            Jacobian calculus            %
  %^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^%
  % XXa   ==> variable state value
  % uT    ==> imput in the actual time
  % uT    ==> imput in the previous time
  % xT    ==> variable state value in the actual time
  % xT_1  ==> variable state value in the previous time
  XXa(:,i)=X0;
  uT=[vd(i) ;vq(i); vfd(i)];
  uT_1=[vd(i-1) ;vq(i-1); vfd(i-1)];
  xT(:,1)=XXa(:,i);
  xT_1(:,1)=XXa(:,i-1);
  %%%%%%%%%%%%%%%%%%%%%%%%
  % jac_rls ==> fuction used to calculate the jacobian in recursive way
  % the impust were defined as  comented before
  % the outputs are:
  % dyT ==> jacobian of the output respect to the estimated parameteres
  % dyT ==> jacobian of the variables states respect to the estimated parameteres
  [dyT,dxT_1] = jac_rls(p,uT,uT_1,xT,xT_1,dxT_1);
  % jacobiano  ==>Time evolution matrix used to plot jacobian matrix
  jacobiano(:,(2*i+1):(2*i+2))=-dyT(:,1:2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                step  6              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            Getting final results          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% H    ==> Hessian matrix
```

```
% ei_v ==> Eigenvalues of H
% P    ==> Final parameter errors
fprintf(' Hessian Matrix \n')
H=jacobiano*jacobiano'
fprintf(' Eigenvalues of H \n')
ei_v=eig(H)
fprintf(' Final parameter errors in %       \n')
for i=1:9
   PE(i)=(P1(i)-p(i))*100/P1(i);
   PEB(i)=(PE(i));
end
PE';
param=['xd       ' ; 'xd`      ' ; 'xd``     '; 'Td0`     ';  'Td0``    ' ; 'k        ' ; 'xq       ';  'xq``     ' ; 'Tq0``    '];
valo=['Parameter ' '       Real Value' '    Initial Value' '  Estimated value' '          % error'];
mm1=[ P1 P0 p PEB'];
RES_1=valo;
RES_1(2:10,1:10)=param;
for i=1:4
   dat_1=num2str(mm1(:,i));
   [ty tx]=size(dat_1);
   RES_1(2:ty+1,(10+17*i-tx+1):(10+17*i))=dat_1;
end
RES_1



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                 step  7                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%               Results plots               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

graficar
```

D.2 Prediction a correction of the parameters and the status (“obs_kal_1.m”) for the IEKF and/or SS

```
% This algorithm correct the state and propagates the kalman gain and the error covariance matrix
%
%^^ Inputs ^^
% p     ==> Parameter vector
% pi    ==> Parameter vector
% YY    ==> Measured output
% X0    ==> propagated state by linear inetgration (linear system)
% PKE   ==> initial error covariance matrix
% W     ==> weighed sensor matrix (related to YY)
% V     ==> weighed estimated parameter matrix (related to p)
% u1    ==> input to the linear system (voltages)
% Ts    ==> sampling time
```

```
% indi2 ==> indices of the parameters to be estimated
%
% ^^ Outputs ^^
% con_p0==> condition number of error covariance matrix P
% EV_P  ==> eigenvalues of error covariance matrix P
% x1    ==> corrected states
% p     ==> estimated parameteres vector
% PKM   ==> propagated and corrected error covariance matrix
% KKK   ==> kalman gain
% e     ==> error betwen the measures and the correted estimated output

function [con_p0,EV_P,x1,p1,PKM,KKK,e] = obs_kal(p,pi,YY,X0,Xi,PKE,PKEi,W,V,u1,Ts,indi2,l,NNN)

% indiT ==> indice  del estado extendido
indiT=[1 2 3 4 5 6 7 8 9 10 11 12];

u=u1(:,2);
xd=p(1);% 1. parameter to be estimated
xd_=p(2);           % 2. parameter to be estimated
xd__=p(3);          % 3. parameter to be estimated
Td0_=p(4);          % 4. parameter to be estimated
Td0__=p(5);         % 5. parameter to be estimated
k=p(6);             % 6. parameter to be estimated
xq=p(7);% 7. parameter to be estimated
xq__=p(8);          % 8. parameter to be estimated
Tq0__=p(9);         % 9. parameter to be estimated

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
% Propagation of the output and calculus of the error betwen   %
% the measured and the estimated output                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%

c12=1/xd__;
c23=1/xq__;
Cx=[0 c12 0;0 0 c23];

d12=-1/xd__;
d21=1/xq__;
Dx=[0 d12 0;d21 0 0];

e=YY'-(Cx*X0+Dx*u);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
% difference H ( output equations) respect to the parameteres    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
Dm=zeros(18,6);
Dm(3,2)=1/(xd__^2);
```

```
Dm(3,5)=-1/(xd__^2);
Dm(17,1)=-1/(xq__^2);
Dm(17,6)=-1/(xq__^2);

CPK=[Dm(1:9,1:6)*[u;X0] Dm(10:18,1:6)*[u;X0] ];
CK=zeros(2,12);
CK(1:2,1:3)=Cx;
CK(1:2,4:12)=CPK';
% Ck ==> C matrix of the extented kalman filter related to the output
% equation

% identifies if it is applied the subset selection technique

PD1=setdiff(indiT,indi2);


if PD1~=0
   % if the subset selection is applied, the matrix Ck and P must be
   % reduced and only is taked the elements wich correspon to the subset
   PKE_R=PKE(indi2,indi2);
   CK_R=CK(1:2,indi2);
   %  the initial error covariance matrix used of IEKF must be reduced
   PKEi_R=PKEi(indi2,indi2);
   % the kalman gain is calculated
   KK_R=PKEi_R*CK_R'*inv(W+CK_R*PKEi_R*CK_R');
   KK=zeros(12,2);
   KK(indi2,1:2)=KK_R;
   % the error covariance matrix is corrected
   PKE0_R=PKEi_R-KK_R*CK_R*PKEi_R;
   % the extended state is generated
   XK=[X0; p];
   % the initial extended state is generated for the IEKF
   XKi=[Xi; pi];
   % the extended state is reduced for SS
   XK_R=XK(indi2,1);
   % the initial extended state is reduced for the SS-IEKF
   XKi_R=XKi(indi2,1);
   % correcting the extended SS state
   XK_R=XKi_R+KK_R*(e-CK_R*(XKi_R-XK_R));
   XK1=XK;
   XK1(indi2,1)=XK_R;
   % updating the extended state
   XK=XK1;
   PKE0=PKE;
   % updating the error covariance matrix
   PKE0(indi2,indi2)=PKE0_R;
   % calculating the condition number of the P
   con_p0=cond(PKE0_R);
   %calculatin the eigenvalues of P
   EV_P=eig(PKE0);
else
```

```
    % calculating the Kalman gain
    KK=PKEi*CK'*inv(W+CK*PKEi*CK');
    % correcting the error covariance matrix
    PKE0=PKEi-KK*CK*PKEi;
    % generating the extended state
    XK=[X0; p];
    % generating the initial extended state for IEKF
    XKi=[Xi; pi];
    % correcting the state
    XK=XKi+KK*(e-CK*(XKi-XK));
    % calculating the condition number of the P matrix
    con_p0=cond(PKE0);
    % calculating the eigenvalues of the P  matrix
    EV_P=eig(PKE0);
end

% updating the state and the estimated parameteres
x1(1:3,1)=XK(1:3,1);
p1(1:9,1)=XK(4:12,1);

% if the maximun number of iteration is equal to the actual iterations
% predict the error covariance matrix (for the IEKF)
 if l==NNN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% diference F ( state propagation equations) respect to the parameteres     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

xd=p1(1);           % 1. parameter to be estimated
xd_=p1(2);          % 2. parameter to be estimated
xd__=p1(3);         % 3. parameter to be estimated
Td0_=p1(4);         % 4. parameter to be estimated
Td0__=p1(5);        % 5. parameter to be estimated
k=p1(6);            % 6. parameter to be estimated
xq=p1(7);           % 7. parameter to be estimated
xq__=p1(8);         % 8. parameter to be estimated
Tq0__=p1(9);        % 9. parameter to be estimated

B11=[0 1/Td0_/xd__ 0 0 -1/Td0_/xd__ 0
0 -1/Td0_/xd__ 0 0 1/Td0_/xd__ 0
0 -(xd-xd_)/(Td0_*(xd__^2)) 0 0 (xd-xd_)/(Td0_*(xd__^2)) 0
0 -(xd-xd_)/(xd__*(Td0_^2)) -k/(Td0_^2) -1/(Td0_^2) ...
(xd-xd_)/(xd__*(Td0_^2)) 0
0 0 0 0 0 0
0 0 1/Td0_ 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0];

B21=[0 1/Td0_/xd__ 0 0 -1/Td0_/xd__ 0
```

```
0 -1/Td0_/xd__+1/Td0__/xd__ 0 0 1/Td0_/xd__-1/Td0__/xd__ 0
0 -(xd-xd_)/(Td0_*(xd__^2))-xd_/(Td0__*(xd__^2)) 0 0 ...
(xd-xd_)/(Td0_*(xd__^2))+xd_/(Td0__*(xd__^2)) 0
0 -(xd-xd_)/((Td0_^2)*xd__) -k/(Td0_^2) 1/(Td0_^2) ...
(xd-xd_)/((Td0_^2)*xd__) 0
0 -(xd_-xd__)/((Td0__^2)*xd__) 0 -1/(Td0__^2) ...
(xd_-xd__)/((Td0__^2)*xd__)+1/(Td0__^2) 0
0 0 1/Td0_ 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0];

B31=[zeros(6)
-1/Tq0__/xq__ 0 0 0 0 -1/Tq0__/xq__
xq/(Tq0__*(xq__^2)) 0 0 0 0 xq/(Tq0__*(xq__^2))
(xq-xq__)/((Tq0__^2)*xq__) 0 0 0 0 xq/((Tq0__^2)*xq__)];

% approximation of the matrix A of EKF by discrete system (Ts)
APK=Ts*[B11*[u;x1] B21*[u;x1] B31*[u;x1]];

a11=1/Td0_;
a12=-(xd-xd_)/(Td0_*xd__);
a21=1/Td0__-1/Td0_;
a22=-(1/Td0__+(xd-xd_)/(Td0_*xd__)+(xd_-xd__)/(Td0__*xd__));
a33=-xq/(Tq0__*xq__);

Ax=Ts*[a11 a12 0;a21 a22 0;0 0 a33];

AK2=zeros(12,12);
AK2(1:3,1:3)=Ax+eye(3);
AK2(4:12,4:12)=eye(9);
AK2(1:3,4:12)=APK';
if PD1~=0
   % For the Subset Selection EKF or IEKF
   % the matrix Ak must be reduced for the SS
   AK2_R=AK2(indi2,indi2);
   % the matrix Vk must be reduced for the SS
   V_R=V(indi2,indi2);
   % Prediction of the error covariance matrix
   PKE0_R=AK2_R*PKE0_R*AK2_R'+V_R;
   PKE0=PKE;
   PKE0(indi2,indi2)=PKE0_R;
else
   % for the IEKF or EKF
   % Prediction of the error covariance matrix
   PKE0=AK2*PKE0*AK2'+V;
end
end
KKK=KK;
PKM=PKE0;
```