1-1-2012

Particle Swarm Optimization Based Adaptive Kalman Filters for Fault Diagnosis of Reaction Wheels

Afshin Rahimi
Ryerson University

PARTICLE SWARM OPTIMIZATION BASED ADAPTIVE KALMAN FILTERS FOR FAULT DIAGNOSIS OF REACTION WHEELS

by

Afshin Rahimi

BSc., Aerospace Engineering

K.N. Toosi University of Technology, 2010

A thesis presented to Ryerson University

in partial fulfillment of the requirements for the degree of

MASTER OF APPLIED SCIENCE

in the Program of

Aerospace Engineering

Toronto, Ontario, Canada, 2012

Author's Declaration

Abstract

Particle Swarm Optimization Based Adaptive Kalman Filters for Fault Diagnosis of Reaction Wheels

Afshin Rahimi, Master of Applied Science, Aerospace Engineering

Ryerson University, Toronto, August 2012

There has been an increasing interest in fault diagnosis in recent years, as a result of the growing demand for higher performance, efficiency, reliability and safety in control systems. A faulty sensor or actuator may cause process performance degradation, process shut down, or a fatal accident. Quick fault detection and isolation can help avoid abnormal event progression and minimize the quality and productivity offsets. In space systems specifically, space and power are limited in the satellites, which means that hardware redundancy is not very practical. If actuator faults occur, analytical redundancy techniques should be employed to determine if, where, and how the fault(s) occurred.

To do so, different approaches have been developed and studied and one of the well-known approaches in the literature is using the Kalman Filter as an observer for the purpose of parameter estimation and fault detection. The gains for the filter should be selected and the selection of the process and measurement noise statistics, commonly referred to as "filter tuning," is a major implementation issue for the Kalman filter. This process can have a significant impact on the filter performance. In practice, Kalman filter tuning is often an ad-hoc process involving a considerable amount of time for trial and error to obtain a filter with desirable –qualitative or quantitative- performance characteristics.

This thesis focuses on presenting an algorithm for automation of the selection of the gains using an evolutionary swarm intelligence based optimization algorithm (Particle Swarm) to minimize the residuals of the estimated parameters. The methodology can be applied to any filter or controller but in this thesis, an Adaptive Unscented Kalman Filter parameter estimation applied to a reaction wheel unit is used for the purpose of performance evaluation of the proposed methodology.

Acknowledgements

After a year and a half and many uncertain and turbulent times, I have finally completed my master's thesis. Thankful to all those who have helped me through this path; My supervisors, Dr. Krishna Dev. Kumar and Dr. Hekmat Alighanbari, have been an invaluable sources of guidance, without which this thesis would not exist. I appreciate the challenges they threw my way and their confidence in me to complete them.

Over these past one and a half years, I came to know other knowledgeable engineers such as the members of the SSDC lab. I appreciate your friendship and camaraderie and learned a lot from each one of you. Hereby I want to thank you individually and let you know that all those memories will never be forgotten. Thanks to Michael who was like an old brother to me. Arthur who taught me how to look at life in order to live happier. Waqas as a hard-working individual with many intuitive ideas. Sobhan whom I learned a lot from as a friend and a younger brother, and the rest of the group that I might not have a word for each but am thankful to every one of them for their support and companion.

I also want to specifically thank my closest friends, Ali, Navid, Mojtaba, Reza, Amirhossein, Ahmad, and Sogol for being there when I needed them whether in sadness or happiness. Regardless of what the future holds, I hope they will always remain a part of my life. One other specific acknowledgement to the ISS (International Students Services) family that supported me and was there for me in all aspects of my life during this period. And also, ISARU (Iranian Students Association at Ryerson University) that exposed me to the Iranian culture in Toronto and within the university. I also need to thank Colleen, Scott, Cherry, Mohsen, Seyyed, Mehrdad and Shahin for all their moral, financial and educational support.

Last but not the least my parents Majid and Farideh, are the ones who their unconditional love made them the most selfless individuals I have ever come to know in my life. I will always be indebted to them and feel blessed and privileged to have them as my parents. This thesis is dedicated to them, in a hope that it would make them happy and proud of the son they raised to this point. Finally, my brother Ramin is someone I need to thank here as he was there like a big brother for me.

For... my loving parents Majid Rahimi and Farideh Farhang without whose support and endless love I would not have achieved this work. And my brother, Ramin, whose support and advice has always been helpful and valuable to me.

Table of Contents

List of Tables

List of Figures

Abbreviations

ACS	Attitude Control System
ADS	Attitude Determination System
AFF	Adaptive Fading Factor
AKF	Adaptive Kalman Filter
AUKF	Adaptive Unscented Kalman Filter
BEMF	Back Electromotive Force
BLDC	Brushless Direct Current
CM	Covariance Matching
CMG	Control Moment Gyroscope
COTS	Commercial Off The Shelf
CT	Continuous Time
DE	Differential Evolution
DT	Discrete Time
EA	Evolutionary Algorithms
EKF	Extended Kalman Filer
FDI	Fault Diagnosis and Identification
FP	Fault Parameter
GA	Genetic Algorithms
HPCVL	High Performance Computing Virtual Laboratory
IAE	Innovation Adaptive Estimation
IMM	Interactive Multiple Model
IMU	Inertial Measurement Unit
ISS	International Space Station
IWA	Inertia Weight Approach
KF	Kalman Filter
LCS	Learning Classifier System
LKF	Linear Kalman Filter
ME	Mean Error
MEMS	Micro-Electro-Mechanical-System
MLA	Maximum Likelihood Approach
MM	Multiple Model
MMAE	Multiple Model Adaptive Estimation
MOA	Magnetic Optimization Algorithm
MOI	Moment of Inertia
MTR	Magnetic Torque Rod
OFCPI	Objective Function Component Performance Index
OS	Operating System
PDF	Probability Density Function
PSO	Particle Swarm Optimization

RAM	Random Access Memory
RMSE	Root Mean-Squared Error
RRG	Robotics Research Group
RW	Reaction Wheel
SAE	Sequential Adaptive Estimation
SI	Swarm Intelligence
SP	Service Pack
SSDC	Space Systems Dynamics and Control
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
ZOH	Zero-Order Hold

Nomenclature

I	identity matrix
$\mathrm{K_m}$	BEMF constant in RW model
T_{cog}	cogging torque in RW model
N	number of motor poles in RW model
ω	rotor speed in RW model
τ_c	coefficient of coulomb friction in RW model
τ_v	coefficient of viscous friction in RW model
K_t	torque constant in RW model
B	ripple torque in RW model
C	cogging torque in RW model
ω_a	torque noise frequency in RW model
θ_a	jitter angle in RW model
K_e	BEMF in RW model
V_{bus}	bus voltage in RW model
G_d	driver gain in RW model
R_{IN}	input filter resistance in RW model
P_q	quiescent bus power in RW model
ω_d	driver bandwidth in RW model
k_f	voltage feedback gain in RW model
J_w	flywheel MOI in RW model
k_s	over-speed circuit gain in RW model
ω_s	maximum wheel speed in RW model
N	number of motor poles in RW model
ω	wheel speed in RW model
i	current in RW model
T_s	sampling period in RW model
P_v	white-noise signal power in RW model
w_{psd}	white-noise power spectral density in RW model
N	Window Size in Kalman Filters
$\mathbf{Q}_{max}$	upper limit for the entries in process noise covariance matrix in AUKF in Kalman Filters
$\mathbf{R}_{max}$	upper limit for the entries in measurement noise covariance matrix in AUKF in Kalman Filters
$\mathbf{R}_0$	initial measurement noise covariance matrix in AUKF in Kalman Filters

$\mathbf{Q}_0$	initial process noise covariance matrix in AUKF in Kalman Filters
$\mathbf{P}_0$	initial noise covariance matrix in AUKF in Kalman Filters
n	dimension of the augmented states in UKF in Kalman Filters
κ	control factor on the spread of the sigma point in UKF in Kalman Filters
$\overline{\mathbf{y}}$	mean of variable $\mathbf{y}$ in Kalman Filters
f	non-linear function in unscented transformation for process model
$\mathbf{x}$	n-state random variable in Kalman Filters
h	non-linear output function in Kalman Filters for measurments model
$\mathbf{W}$	component weight for covariance calculation in Kalman Filters
$\hat{\chi}$	initial sigma points in Kalman Filters
$\tilde{\chi}$	instantiated sigma points through process model in Kalman Filters
$\mathbf{Q}$	process noise covariance matrix in Kalman Filters
$\mathbf{R}$	measurement noise covariance matrix in Kalman Filters
$\mathbf{P}_{xx}$	covariance of states matrix in Kalman Filters
$\mathbf{Y}_i$	instantiated sigma points through measurement model in Kalman Filters
$\mathbf{P}_{yy}$	covariance of measurements matrix in Kalman Filters
$\mathbf{P}_{xy}$	cross covariance matrix in Kalman Filters
$\mathbf{K}$	Kalman filter gain
α	scale-factor for error covariance prediction and/or gain calculations in Kalman Filters
$\hat{\mathbf{C}}_v$	estimated innovation covariance matrrix in Kalman Filters
$\mathbf{C}_v$	theoretical innovation covariance matrix in Kalman Filters
$\hat{\mathbf{C}}_r$	estimated residual sequence covariance matrix in Kalman Filters
$\mathbf{r}_{k+1}$	unbiased estimator for residual in Kalman Filters
H	Jacobian of h in Kalman Filters
$\hat{\mathbf{R}}$	estimated measurement noise covariance matrix in Kalman Filters
$\hat{\mathbf{Q}}$	estimated process noise covariance matrix in Kalman Filters
$\Delta\overline{\hat{\mathbf{x}}}$	mean of the innovation in Kalman Filters
$\mathbf{x}_a$	augmented state-vector in Kalman Filters
ε	fading factor in AUKF in Kalman Filters
ζ	adaptive fading factor/fogetting factor in AUKF in Kalman Filters
X_i	position vector of the ith particle in the swarm in PSO
P_i	position history vector of the ith particle in the swarm in PSO
V_i	velocity vector of the ith particle in the swarm in PSO
$\mathrm{p}_{\mathrm{best}}$	best position in the history of a particle in PSO
$\mathrm{g}_{\mathrm{best}}$	best position in the history of the swarm in PSO
w	inertia factor for velocity in PSO
c_1	coefficient of the self-recognition component in PSO

c_2	coefficient of the social component in PSO
r_1	random number to diversify the self-recognition component in PSO
r_2	random number to diversify the social component in PSO
w_{max}	initial weight in inertia factor for velocity in PSO
w_{min}	final weight in inertia factor for velocity in PSO
$iter_{max}$	maximum number of iterations in PSO
$iter$	current iteration number in PSO
X_i^+	new position for the ith particle in the swarm in PSO
X_i^-	old position for the ith particle in the swarm in PSO
J	objective/index function in PSO
r_j	normalizer component for the objectvie function in PSO
e_k	error (difference between estimated and actual value for the parameter)

1. Introduction

The procedure of selecting the process and measurement noise covariance matrices components, commonly known as “filter tuning” is a major implementation issue for the Kalman filter. This process can have significant impact on the performance of the filter. Since this is an ad-hoc process involving a considerable amount of time for trial and error to obtain the desirable performance characteristics. Motivated by that, the focus of this thesis is to propose a methodology using optimization algorithms to automate this process and the make human factor and interference impact in the tuning process minimum.

The term “mathematical optimization”, which also is known as optimization or mathematical programming, in mathematics, computer science, or management science, refers to the selection of a best element from some set of available alternatives. In the simplest case, an optimization problem consists of maximizing or minimizing a real function by “systematically” choosing input values from within an allowed set and computing the value of the function. The generalization of optimization theory and techniques to other formulations comprises a large area of applied mathematics. More generally, optimization includes finding "best available" values of some objective function given a defined domain, including a variety of different types of objective functions and different types of domains.

In the field of optimization, Fermat [1] and Lagrange [2] found calculus-based formulas for identifying optima, while Newton [3] and Gauss [3] proposed iterative methods for moving towards an optimum. Historically, the first term for optimization was "linear programming", which was due to George B. Dantzig [4], although much of the theory had been introduced by Leonid Kantorovich, a Russian mathematician, in 1939 [5]. Dantzig published the Simplex algorithm in 1947 [4], and John von Neumann developed the theory of duality in the same year.

Linear programming arose as a mathematical model developed during World War II to plan expenditures and returns in order to reduce costs to the army and increase losses to the enemy. It was kept secret until 1947. Postwar, many industries found its use in their daily planning.

In 1963, Dantzig’s Linear Programming and Extensions was published by Princeton University Press. Rich in insight and coverage of significant topics, the book quickly became "the bible" of linear programming. Since then, the optimization approaches and methodologies have advanced and been used in many different subjects. The evolutionary algorithms nowadays are the main field of interest for the researchers and the industry as they can be used to solve problems that are more complex.

1.1 Motivations

The current research on the Fault Diagnosis, Identification, and Isolation (FDI) for space systems is not usually incorporated with the optimization algorithms. In the FDI procedure, Kalman filter is one of the most commonly used tools for filtering and parameter estimation but there is one major issue with it and that is how to tune the filter systematically and not by trial and error. Tuning the filter is a tedious task and the performance of the tuned filter is tremendously dependant on the experience of the person who is doing the manual tuning.

In the literature, different attempts to implement the optimization algorithms in order to remove the human impact on the performance of the filter by automating the process are available. In Ref. [6] auto-covariance least-square estimation is proposed for estimating the noise covariances from process data. In Ref. [7], a new approach is presented to estimate the parameters using particle swarm optimization but this approach is not using PSO to tune the filter and it is proposing a new filter using the particle swarm optimization method. In addition, in Ref. [8] another method based on Genetic Algorithm is proposed to estimate fate and transport parameters of a reacting solute from the column and batch experiments involving a saturated porous medium. In Ref. [9] the attitude estimation method of humanoid robot using an extended Kalman filter with a fuzzy logic based tuning algorithm is presented. Ref. [10] investigates an application of a 'discrete variable' hybrid differential evolution (dvHDE) method to parameter estimation of a single wheel station, which incorporates dvHDE and Kalman filter. This reference also compares the results with the standard gradient-based (GB) method, Downhill Simplex (DS) method and the original differential evolution (DE) method on simulated and experimentally obtained data. Ref. [11] investigates the application of PSO in tuning the UKF covariance matrices. In addition to that in another conference publication by the authors in Ref. [12] a comparison between PSO tuned and another optimization algorithm known as Bacterial Foraging Optimization (BFO) is conducted. In Ref. [13], tuning of Extended Kalman Filter (EKF) using PSO is investigated and a comparison between PSO tuned and the conventional

EKF is presented. Ref. [14] proposes a self-tuning Kalman filter applied to engine control systems. A discussion on different algorithms used for the tuning of Kalman filters is give in Ref. [15]; the authors then propose a new algorithm called Local-to-Best Mutation with Shuffled Steps (L2BM-SS-DE) variation of Differential Evolution. In Ref. [16], the author discusses on different issues with the tuning of a Kalman filter with different approaches and then uses the Simplex algorithm for the tuning.

Reviewing all the above-mentioned literature and motivated by the issues faced in Ref. [17], including no systematic tuning approach and hence, the results are not guaranteed to be the best possible, at least within a range of parameters, in performance. In addition, because the tuning is done manually, the process is very time consuming and for each simulation, the parameters need to be selected separately. For different noise levels, the filter needs to be set again and the performance varies from one case to another. Moreover, for different systems, the tuning should be done manually. This task is tedious considering that it requires the user who is tuning the filter to have enough knowledge and experience about the system to be able to tune it practically.

The purpose of this research is to apply an evolutionary optimization algorithm known as Particle Swarm Optimization to automate the process of filter tuning and by that solving the issues mentioned above and as were faced in the SSDC lab during the process of development of the previous FDI algorithm in Ref. [17]. Then analyze the outputs of the simulations to further investigate the applicability of the proposed methodology. In addition to that, it was interesting to study whether or not this methodology could be applied to other systems. Moreover, the fact that this methodology was required to be easy to use and modular to make it adaptable to other applications was another challenging issue to be solved.

1.2 Reaction Wheel as an Actuator for Satellite

Reaction Wheels have been the most common attitude control technologies used as actuators that can provide full 3-axis attitude control for the small satellites with meaningful missions in Earth. A RW consists of a flywheel mounted to an electric motor. RWs have been used extensively for active control of spacecraft. Generally, RWs can perform slow manoeuvring of satellites with average slew rates of approximately 1 deg/sec to avoid saturating the wheel and keeping power consumption low. Saturation occurs when the maximum wheel speed is reached and no more actuation is available in the direction of increasing wheel speed, this phenomenon is usually resolved using momentum dumping techniques. Power consumed during a manoeuvre is directly related to the reaction torque generated by a RW, fast slews translate to larger motor torques and consequently larger current draws.

For RW torque generation a transient current occurs when a voltage is applied at the armature until the back EMF (BEMF) voltage amplitude comes close to matching the applied voltage with opposite polarity, this results in a near zero voltage across the armature circuit so long as the wheel speed or applied voltage does not change. More current is drawn in a transient state of the wheel speed than at constant speed, and RW actuation is the result of changes in wheel speed.

1.3 Fault Diagnosis and Identification

During the entire mission life of a satellite costs are accrued and faults/failures add to it, if occur. The most probable area for a fault to occur is the satellite attitude control system (ACS). Actuators usually consist of moving mechanical parts subject to unforeseen faults or failures such as a cold solder joint affecting electrical performance, minute particles interfering with mechanical components, or massive temperature fluctuations. As a result, FDI techniques have been developed so that some of these factors can be monitored and/or predicted.

Fault diagnosis refers to the detection and isolation of faults, and identification deals with the type of fault and its severity. In the past with larger satellites, faults and/or failures were handled through hardware redundancy because there was more onboard space and consequently, online computing power was at a premium. In the current state, hardware redundancy, in order to reduce costs, is limited. In case a fault occurs, numerical methods can be utilized to diagnose and correct the problem(s). If the satellite is able to detect, isolate, and identify these faults then operators on the ground can move quickly to obtain the best possible performance from the satellite, or the satellite could simply correct the problem itself. Ultimately, the goal is to facilitate at least partial completion of a mission in case of faults or failures.

Since RWs are very common actuators, online FDI algorithms should be available that can monitor them. A “fault” is used to denote an unpredicted change of system behaviour that results in a degradation of performance or prevents any semblance of normal operation of the system. They can be classified based on their duration and severity (Figure 1.1).

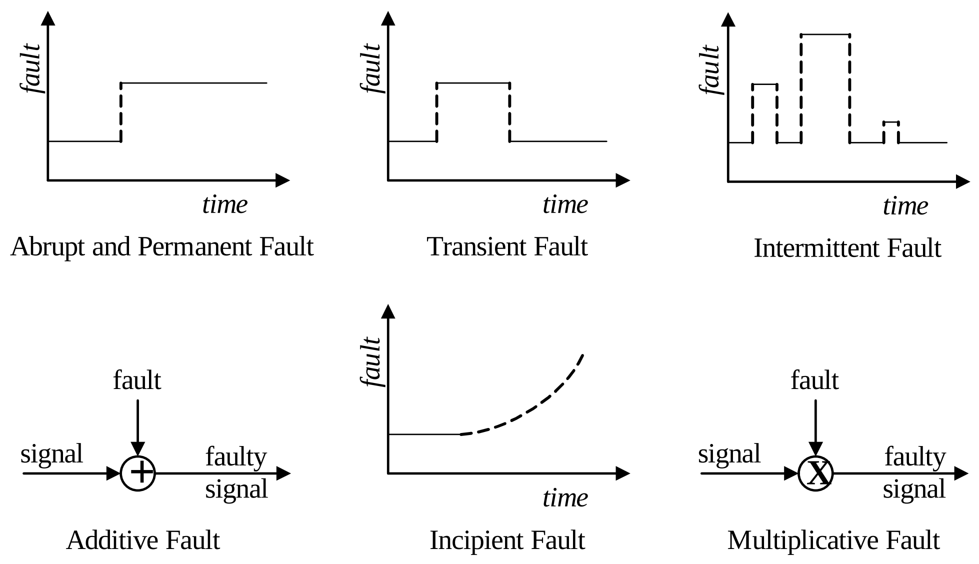

Figure 1.1 Types of Faults

Faults can occur suddenly or slowly over time. Sudden faults are usually caused by external disturbances that severely damage a component. Once they occur, replacement of the faulty component is the best course of action because the unexpected nature of the fault can introduce large stresses to the structure or electronic system. Incipient faults occur slowly over time such as increased frictional losses in ball bearings. In this case, immediate replacement of the faulty component is not necessary as long as satisfactory performance can be achieved. Transient and intermittent faults occur randomly for bounded periods and then vanish, this makes them difficult to identify. They differ with respect to the number of states they can assume. For example, a transient fault assumes one state and then goes back to normal, whereas intermittent faults can assume various states throughout a faulty time-interval. Additive faults are simply superimposed on healthy signals of a system, while multiplicative faults are directly proportional to system states [17].

As discussed earlier, there are three primary components to fault diagnosis algorithms 1) detection, 2) isolation, 3) identification.

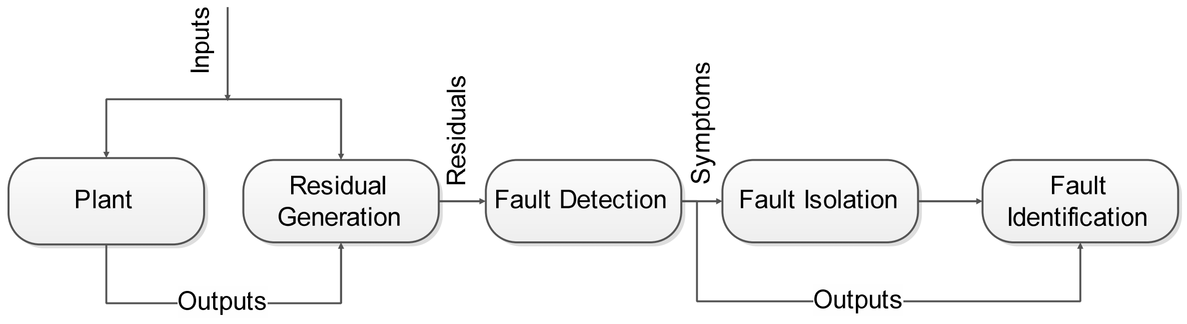

Figure 1.2 Steps in Fault Diagnosis

The scheme should be able to detect the occurrence of a fault, isolate where the fault has occurred, and for completeness identify what type of fault has occurred as shown in Figure 1.2. These tasks are not relevant for all applications; some applications will only require fault detection while others will also require isolation. Detection would only be required when regardless of the type of fault or its location the corrective action is the same, in this case the algorithm or mechanism would simply alert operator that a fault has occurred. Isolation could be required in a complex system with many moving parts that could not be examined precisely by an operator due to the size and complexity of the system. In this case, the algorithm should be able to detect and isolate the location of a fault so that the corrective action is fast and efficient. Identification of a fault is not always necessary when an operator is in charge of the corrective action; however, it is an important element to fault diagnosis if the operators need to know right away what component faulted and why so that they could determine right away what type of corrective action to take specifically for that component. For autonomous systems, the operator can be considered the computer that makes decisions based on data processing algorithms, otherwise the operator can be considered human. FDI algorithms usually rely on sensor measurements data to determine the health of the system and what type of corrective action to take. It is widely understood that sensors are subject to various errors and real-life systems are subject to disturbances, however false alarms due to measurement noise and system disturbances must be avoided as much as possible. Residual generation can be approached using one of three frameworks shown in Figure 1.3. The two main steps of an FDI algorithm are 1) extraction of trends from measurements, 2) the interpretation of the extracted trends [18]. Extensive work has been done on trend extraction and representation including the establishment a formal framework for representing process trends [19].

1) Mathematical model-based: Model-based fault diagnosis consists of using a mathematical model of the system in question to describe the ideal behaviour of the system in a fault-free case. The output of the model is used to perform consistency checks against the measured states of the system; if the measurements deviate noticeably from the expected system behaviour then it is assumed that the system is operating in a faulty mode. The consistency check involves taking the difference between the model output and system measurements. This results in a *residual* error signal that is vital to the proper design of any FDI algorithm. Residual signals should be zero in the healthy case, however in practice, measurements are subject to white noise and systems are subject to disturbances, which limits the residual to be in the neighbourhood of zero in healthy modes.

The problem with using signal-processing techniques is that they do not consider the dynamic relationships between the measured signals of a system. The processed data represents a system as a whole with disturbances, non-linearities and the normal system operation being combined into one response. The ARMAX model is a general deterministic-stochastic model developed in 1978 [20] to extract information about the system dynamics and noise.

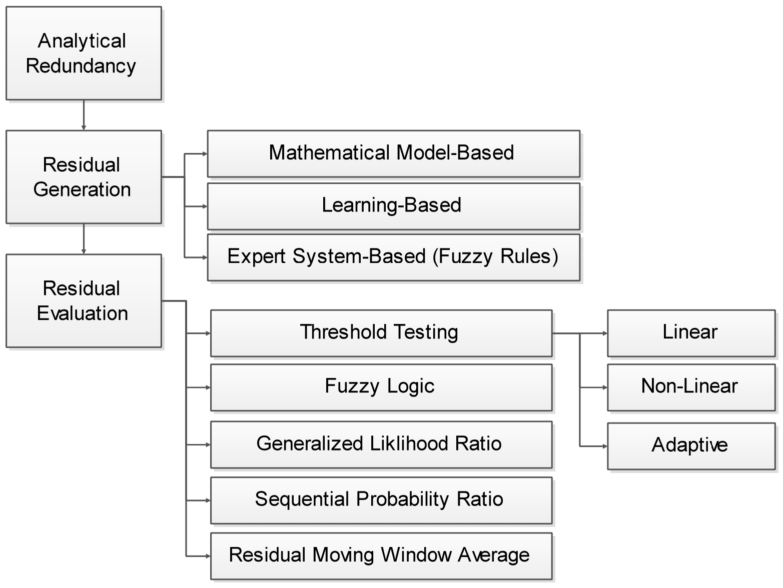

Figure 1.3 Analytical Redundancy Techniques

Mathematical model-based residual generation requires a mathematical model of a system derived using physical principles. Model-based schemes can be classified further into observer-based methods that employ linear or non-linear observers [21, 22] in a deterministic setting and Kalman filtering [23] (extended Kalman filter (EKF), Unscented Kalman Filter (UKF), Adaptive Kalman Filter (AKF), and the linear Kalman filter (LKF)) in a stochastic setting. Residual signals are defined as output estimation error and innovation sequence for deterministic and stochastic settings respectively. The parity-space method is based on simple algebraic projections and geometry that provide an appropriate check of the consistency of system measurements [24]. This method has been applied to both linear and non-linear systems [25] for fault diagnosis and is sensitive to measurement and process noise relative to observer-based methods. Another approach to residual generation is parameter estimation. The goal here is to estimate physical system parameters, a change in any given parameter estimate will provide information as to where the fault occurred and what kind of fault occurred and the severity. Parameter estimation methods such as continuous-time adaptive parity equations [26], neural parameter estimators [27], Kalman filtering, Neural Networks, and least squares [28] have been applied for FDI purposes.

The FDI for thrusters has been designed by Boskovic et al. [29] based on the concept of Multiple Models, Switching, and Tuning (MMST) while a nonlinear iterative neuron proportional, integral, and derivative (INPID) observer based FDI was developed by Wu and Saif [30]. The parameters of the observer input were updated based on the proportional, integral, and derivative information of the fault estimation error. On the other hand, the FDI for reaction wheels (RWs) has been developed by Tudoroiu and Khorasani [31] using interacting multiple model Kalman filters while Azarnoush and Khorasani [32] applied linear and non-linear observers for residual generation and threshold testing for residual evaluation. The main deficiency of adaptive-observer-based fault identification is their inability to detect and estimate time-varying faults. To overcome this limitation, an iterative learning observer (ILO) was proposed by Chen and Saif [33].

2) Learning-Based (computational intelligence-based): Learning-based algorithms use time-histories of the input/output data of a system to learn the system model. This learned model is then used to generate the residuals. This approach can be very appealing when a high-fidelity model of a system is not available or is difficult to obtain. However large quantities of input/output data must be available that encompass most of the expected system behaviour in order for these methods to yield good results. References [34-36] provide comprehensive surveys of learning-based methods, also referred to as computational intelligence-based methods,

artificial intelligence-based methods, soft-computing approaches, or intelligent methods.

Li and Khorasani [37] proposed a FDI algorithm using a dynamic neural network to learn the non-linear RW dynamics and generate residuals that are evaluated by threshold testing. Later, the fault identification function added to FDI, called FDD, was developed using a mathematical model of a system known a priori along with self-learning computational intelligence techniques [38]. However, the algorithm can only identify one fault at a time and its application to different systems requires redesign of the intelligent portion of the algorithm and re-training.

3) Expert System-Based (fuzzy rule-based): A fuzzy logic or neuro-fuzzy system will employ a set of "*if the*" rules that are based on expert knowledge of the system. If this expert knowledge is not available then neural networks are employed to determine the parameters of those rules. Any one of these methods is sufficient for residual generation and has been studied extensively.

Neural networks are well suited to performing trend analysis and/or feature extraction to determine the operating state of a system [39] as well as fuzzified symbolic representations [40]. Clustering algorithms based on Bayesian classification rules have also been developed to classify data into clusters that are centered about means determined a priori; each mean represents a particular operating condition of a system [41]. Interpreting the classified data has been done using alignment-based sequence-matching algorithms [42], hidden Markov models [43], and dynamic time warping for similarity estimation [44]. Frequency domain analysis of a time-series of measured system states and outputs is another form of signal processing. The Discrete Wavelet Transform has been used for over fifteen years to perform feature extraction as fault diagnosis for machinery components [45]. The DFT algorithm transforms data from a time-domain representation to its frequency domain representation where its frequency response can be analyzed and interpreted [46].

The next stage in the FDI problem is residual evaluation. With a properly constructed residual, the healthy and faulty modes of operation should be manifested in the residual such that each mode creates a unique residual signal or pattern. The residual evaluation stage involves processing the residual signal such that its patterns can be matched to particular system modes of operation. A threshold test of instantaneous values, calculating moving window averages of the residuals, or more complex statistical methods such as generalized likelihood ratio or sequential probability ratio testing.

The category of threshold testing involves establishing upper and lower thresholds about the residual mean in the healthy system mode of operation. If the residual signals exceed these boundaries then a fault has been detected. Reference [47] does fault detection for a RW with linear and non-linear observers for residual generation and threshold testing for residual evaluation. In Ref. [48], a simple threshold test of brushless DC (BLDC) motor currents is used to detect faults. A fault detection and isolation algorithm is presented in [37] by applying a dynamic neural network to learn the non-linear RW dynamics and generate residuals that are evaluated by threshold testing. Adaptive thresholds that adjust based on system inputs have also been studied in [49], however this still only provides the ability to detect and in some cases isolate faults.

When residual evaluation is performed using moving window averages of the residuals, the statistics of the residuals are used to estimate system measurement and process noises. This methodology has been applied extensively to the KF in the form of AKFs that adapt the measurement and process noise on the system using this fading memory technique. Reference [50] uses the residual window averaging technique for attitude determination with Global Navigation Satellite Systems (GNSS). And [51] uses this technique for state estimation of non-linear industrial systems., and [52] uses residual window averaging to estimate measurement and process noise for combining a low-cost inertial measurement unit (IMU) with GPS readings to obtain positioning and attitude information.

Finally, statistical methods can be considered for residual evaluation. The Generalized Likelihood Ratio Test (GLRT) [53] is a useful tool in detecting changes in a residual. This method computes a threshold based on the probability of false alarm and corrects detection thus making it more robust than a fixed threshold algorithm. Moreover, it can estimate an abrupt jump in residual amplitude and the time of the jump. However, the proposed method does not work if the residual change is not abrupt, and there are robustness issues against modeling errors and process disturbances. The sequential probability ratio test (SPRT) assumes that samples are uniformly distributed and independent, and that the structure of their distributions is known a priori with unknown parameters. In Ref. [54], the Mann-Whitney rank sum test is used so that the PDF of the samples does not need to be known a priori. However, this method is also not robust to modeling errors and large process disturbances. Both these methods use residual threshold testing based on statistical properties and probabilities of the system in question to improve upon the fixed threshold technique.

In the literature there are many fault diagnosis techniques that have been developed such as Bayesian classification for fault detection and isolation [55], Wavelet and dynamic recurrent neural networks for fault detection and isolation respectively [56], fuzzy neural networks for fault detection [57], learning-based diagnostic tree approaches for fault detection and isolation [58], the Interactive Multiple Model (IMM) approach to detect and isolate faults for reconfigurable control, and adaptive observer methods. Most methods do not perform fault identification however. Reference [18] is one of the few that can perform fault detection, isolation, and identification of RWs. This is done utilizing a mathematical model of a system known a priori along with self-learning computational intelligence techniques resulting in a hybrid approach. Computational loads of this algorithm are large and its application to different systems requires redesign of the intelligent portion of the algorithm, for satellite ACS FDI an approach is proposed in [17] that is claimed to be able to be applied to various actuators without much effort and large computational loads. However, as mentioned earlier in this Chapter there are some disadvantages to this methodology, which in this research an attempt is made to solve the mentioned problems, and investigate the applicability of the proposed algorithm and methodology.

1.4 Problem Statement

In the sections above, an overview of current trends and studies in the field of tuning a filter as well as a literature review on the FDI and different applications for, particularly in space, and RW units was presented.

It is clear that there is a need for a methodology to solve issue with the manual tuning of filters/controllers. Reason being that the process is tedious considering the amount of time it takes a person to do the trial and error with the fact that the experience of the person who is doing the tuning affects the performance of the filter tremendously. Motivated by that there were couple of problems faced in the SSDC lab with the research existed that are addressed below:

[Prob1] *Auto Filter Tuning Methodology*: As mentioned above, there was a need for a systematic approach to tune filters that lacked existing in the SSDC lab for tuning Kalman Filters in Ref. [17]. The problem with the algorithm in Ref. [17] is that it requires a person to tune the filter manually for any sort of change in the simulation including noise levels, system parameters, etc.; which takes a lot of time and does not guarantee the performance of the filter as it is not known by the tuner (person) whether there would be a better solution.

[Prob2] *Covering Different Noise Levels*: Another problem that was faced in Ref. [17] was that for different noise levels in the system there was a need for re-tuning the filter to get “satisfactory” results and again the issue is that the results cannot be guaranteed to be the best as they are just tuned manually and the performance of the filter depends on the experience of the designer/tuner of the filter solely.

[Prob3] *Applicability of use for Different Systems with Minimum Changes Required*: Another requirement for a practical methodology is its applicability to different systems with unknown parameters while promising a satisfactory performance. This was another challenge in the methodology proposed in Ref. [17], which is caused by the fact that the tuning is done manually in that reference. Also the structure of the code in the simulation was not “modular” meaning that the user needed to go through lines of codes to be able to change a little property of the system and then this change should be done in different places including a MATLAB m code and a Simulink model with the effort of running them separately and plotting the results and analysing them afterwards.

1.5 Research Objectives

As per the problems mentioned in the previous section, this research focuses on proposing a new approach for tuning filters (specifically) or any controller in general application which is automated and systematic and gives you certain indices of its performance based on optimization algorithms.

Design of a Tuning methodology based on Optimization: The objective of this dissertation is to propose a methodology, which is able to tune different controllers/filters based on a certain measure and gives different indices of its performance to guarantee the convergence of the solution and the trend of the objective set in advance. This methodology should be robust to changes and not dependant on the system it is being applied to. It also should be structured in a way, which is easy to use and easy to adapt to different systems without sensitivity to changes in the system parameters and system unknowns. If required, it should be able to handle multi noise/disturbance levels in the system and give gains/parameters that would work for different levels of noise/disturbance.

The proposed methodology is applied to an FDI algorithm with Adaptive Unscented Kalman filter (AUKF) for parameter estimation of RW [17]. Its performance is verified via MATLAB/Simulink simulations using a high fidelity RW model [59]. As mentioned before the tuning problem has already been addressed in the literature but in here the problem is applied to a specific system with the scope of structuring the algorithm for easy adaptation to other systems. It is also an important factor to mention that because of the nature of such systems/simulations, there are some changes needed in the structure of the optimization algorithm as well as the structure of the whole methodology to handle divergence of the simulations and yet give the optimum solution within the area of search set in advance.

1.6 Main Contributions

The main contribution of this work is proposing a methodology to tune different filter/controllers automatically with the least human interference so that the results are robust and guaranteed to be near optimal in the search region. The methodology proposed in here is simple and intuitive in nature, it is based on an evolutionary swarm intelligence optimization algorithm, which mimics the behaviour of a group of birds in an attempt to find food as a group. The optimization method is known as Particle Swarm Optimization (PSO) [60] and a variant of this optimization method is used in this thesis to make it bounded and able to handle constraints in an intuitive way. The objective in the PSO for this specific problem would be to minimize the error in the measurements and give satisfactory estimates of the system parameters. This is done by evaluating residuals for different sets of the parameters/gains for the filter and among them deciding on which is the best based on the objective function. The objective function could be set for different problems to adapt the system and objective. In the following Chapters, more details are given on how this is done and how to implement the method in different problems for different systems.

This algorithm is applicable to any general linear or non-linear system. A nonlinear system is demonstrated in this thesis. In particular, the proposed methodology is applied to an AUKF variant [17] for an FDI problem on a high fidelity RW model [59].

1.7 Thesis Outline

In Chapter 2 satellite actuators and in particular, the RW model is explained to understand the system better, where the FDI algorithm is applied to. In Chapter 3, an extensive review of KFs with more focus on unscented Kalman filter and adaptive filters with their applications to FDI problems is presented. In Chapter 4, a review on optimization algorithms is presented and Particle Swarm Optimization (PSO) algorithm is explained in detail. Furthermore, the proposed methodology is explained in detail followed by an example for better understating the optimization process. Different parameters effect on the performance of the proposed methodology is then investigated by different simulation cases presented in Chapter 5. Eventually, Chapter 6 contains the conclusions and future work.

2. Reaction Wheel Model

The focus of this thesis is the tuning technique for a filter/controller but the proposed methodology needs to be applied to a system and the performance needs to be evaluated. Hence, the system that the proposed methodology is going to be applied to is a RW model [59] to have a measure for comparison with the results in Ref. [17]. Consequently, the model being used is the same model from Ref. [17] to give a perspective on how the performances vary/improve with the new methodology applied for tuning the filter.

The ACS of any satellite regardless of class whether active or passive is one of the most critical subsystems for the successful completion of a mission. Passive attitude control consumes no satellite resources and usually takes advantage of the Earth's gravitational and magnetic fields. Active attitude control consumes either fuel or electrical power or both to maintain a desired attitude. The choice of which method to use depends on the required pointing accuracy for the proposed mission. The deciding factors for pointing accuracy are the communications system and type of payload(s) [61]. Assuming an antenna with a beam-width of 1° on a satellite in a 600 km LEO, a simple trigonometric calculation indicates the beam would cover a $10\,km^2$ area on the Earth. Thus in order to guarantee continuous communication with the ground station the satellite must be capable of pointing its antenna with an accuracy of better than 0.5°. In practice, the beam width of a directional antenna is usually wider than 1°. However, it is evident that a less directional antenna requires less pointing accuracy because it can cover more surface area on the Earth. The same argument can be applied to imaging and other payloads that require some degree of pointing accuracy.

To date, RWs and magnetic torque rods (MTRs) are the most common actuators that can fit within small satellites mass, power, and volume constraints and still provide adequate attitude control performance [17]. These constraints vary depending on the payload and mission requirements. This section is a brief description on the satellite actuators with a focus on RWs.

2.1 Actuators

The most common actuator technologies for cubesats are reaction wheels (RWs), thrusters, and magnetic torque rods (MTRs), while CMGs are also common but in larger satellites. A RW consists of a flywheel attached to an electric motor. At least three RWs mounted orthogonally about each of the body axes are required for full three-axis attitude control. When the satellite must perform a maneuver the RWs accelerate and impart a torque onto the spacecraft, if a spacecraft must maintain a desired attitude in the face of external disturbances the RWs must absorb any added momentum to keep the total angular momentum of the system at zero. An MTR usually consists of a wire coil with a ferrite core. When a current passes through the coil a magnetic field is created. When multiple MTRs are combined a magnetic dipole can be created that counteracts the Earth's magnetic field and provides two-axis pointing of a spacecraft. MTRs are usually employed along with a reaction wheel for full three-axis attitude control. Finally the CMG is another momentum exchange device like the RW. A CMG unit consists of a flywheel that is gimballed about one, two, or three of its axes. Gyroscopic torques are generated as the angular momentum vector is rotated about axes perpendicular to the flywheel spin-axis.

Depending on mission requirements one actuator may be more appropriate than others. in particular, the required degree of pointing accuracy is a primary factor for selecting actuators. MTRs provide the lowest pointing accuracy because of the time-varying nature of the Earth's magnetic field and their inability to provide control about more than two axes. They are usually used in conjunction with RWs and CMGs for momentum dumping. For a cubesat MTRs are small enough to satisfy the mass/power/volume constraints hence making them a popular choice. RWs provide substantially improved pointing accuracy and agility relative to MTRs, however power consumption and mass tend to be larger. CMGs provide more accurate and agile pointing capabilities because of their torque amplification characteristics and gyroscopic stabilization. It must be noted that pointing accuracy is a only as good as the combined ADS and ACS(ADCS) accuracy. In other words if the ADS is only accurate to 1 deg and ACS pointing accuracy to 0.5 deg, no better than 1 deg pointing accuracy can be achieved and vice-versa.

2.2 Reaction Wheels

Fundamentally a RW (Figure 2.1) is a flywheel mounted to an electric motor. Electric Motors in space are usually BLDC or stepper motors as opposed to brushed motors. These types of motors are preferred because brushes can scrape particulate matter of the electrodes and contaminate instrumentation. A motor consists of stationary stator windings, and a permanent magnet or wound rotor. The difference between brushed and BLDC motors lies in the commutation method. BLDC motors commutate using stationary position sensors located as close to the rotor magnets as possible. The position sensors are Hall effect sensors that output a logic high level when a magnetic field is passing over them and low when no field is present. Each position sensor generates a pulse-train that is 120 deg out of phase with the other two sensor signals. Commutation is performed by processing the three signals and knowing when to excite a particular stator winding. The rotor speed can also be ascertained by observing the position sensors signal frequencies. In contrast a brushed DC motor uses metallic or carbon conducting 'brushes' to commutate while the stator remains similar to that of the BLDC. The commutator is usually located above the stator windings so that as the motor turns the brushes slide over the commutator making contact with the different commutator segments. Each segment is attached to one winding resulting in the generation of a dynamic magnetic field inside the motor when a voltage is applied across the brushes. This field repels the rotor magnets or windings resulting in the rotation of the rotor. A major problem with brushed DC motors is the wear and tear on the brushes and commutator, in the vacuum of space the tiny particles that wear off of the brushes can disperse in all directions and contaminate on-board electronics.

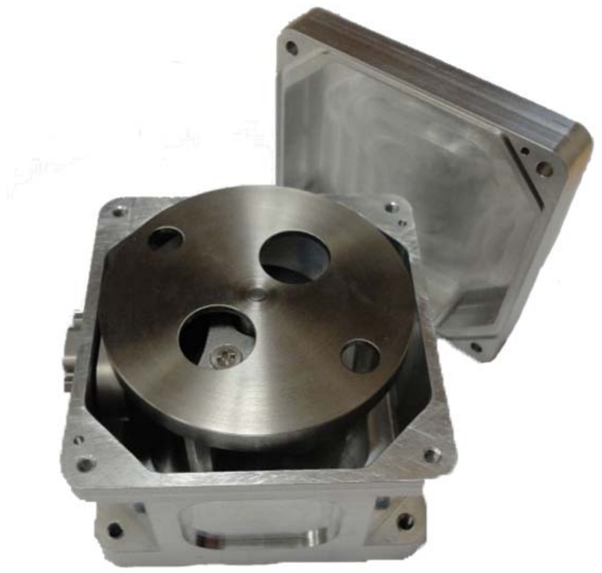

Figure 2.1 Reaction Wheel unit developed in the SSDC lab at Ryerson University

A RW model must consider motor disturbances, non-linearities, and BEMF torque limiting. Figure 2.2 is a high-fidelity RW model for a torque-controlled BLDC motor developed by [59]. Voltage-controlled motors share the same disturbances and non-linearities. BEMF voltages are generated in stator windings when the rotor rotates. A faster rotor speed will yield a larger BEMF voltage, its exact value is determined by the product of wheel speed and BEMF constant K_m, with SI units rad/s and $V/rad/s$ respectively. In so far as torque limiting, when a voltage is applied to the motor the rotor rotates. It will rotate until a speed is reached at which the BEMF voltage is close to the applied voltage such that the differential voltage across the armature is small resulting in a small current. Rotation stops when the current is so small that the motor does not generate enough torque to accelerate. For example if five volts are applied at zero wheel speed the wheel will accelerate until the BEMF voltage nears five volts. In order to decelerate the wheel a lower voltage must be applied and vice-versa to accelerate the wheel again.

On the mechanical side of the dynamics the motor can be subject to disturbances such as cogging and ripple torque. Cogging torque is caused by the rotation of the magnets in the rotor with respect to the motor windings. As a magnet rotates past a winding, its motion is first opposed by flux leakage from the end of the windings until it passes over the entire winding when the motor is then accelerated by the flux leakage. With current BLDC motor technologies cogging torque is no longer a concern as most designs minimize the amount of ferrous material rotating across the windings.

Equation (2.1) describes the cogging torque mathematically,

$$T_{cog} = B\sin\left(3N\omega t\right) \tag{2.1}$$

where B is a gain, N is the number of motor poles, and ω is the rotor speed. Torque ripple occurs at the commutation frequency and is characterized as a variation in the motor torque caused by the commutation method and the shape of the BEMF waveform. For analytical purposes this disturbance is approximated as a sinusoid while in reality it is closer to a truncated rectified sine wave. The equation for this disturbance is shown below,

$$T_{rip} = C\sin\left(\frac{N\omega t}{2}\right) \tag{2.2}$$

where C is a constant and the other parameters are the same as in Eq. (2.1). BLDC motors are also subject Coulomb and viscous friction non-linearities that are dependent on the bearing material and lubricant. Coulomb friction is caused by the rolling friction within the bearings and is characterized by a torque discontinuity when the motor is not generating enough electrical torque.

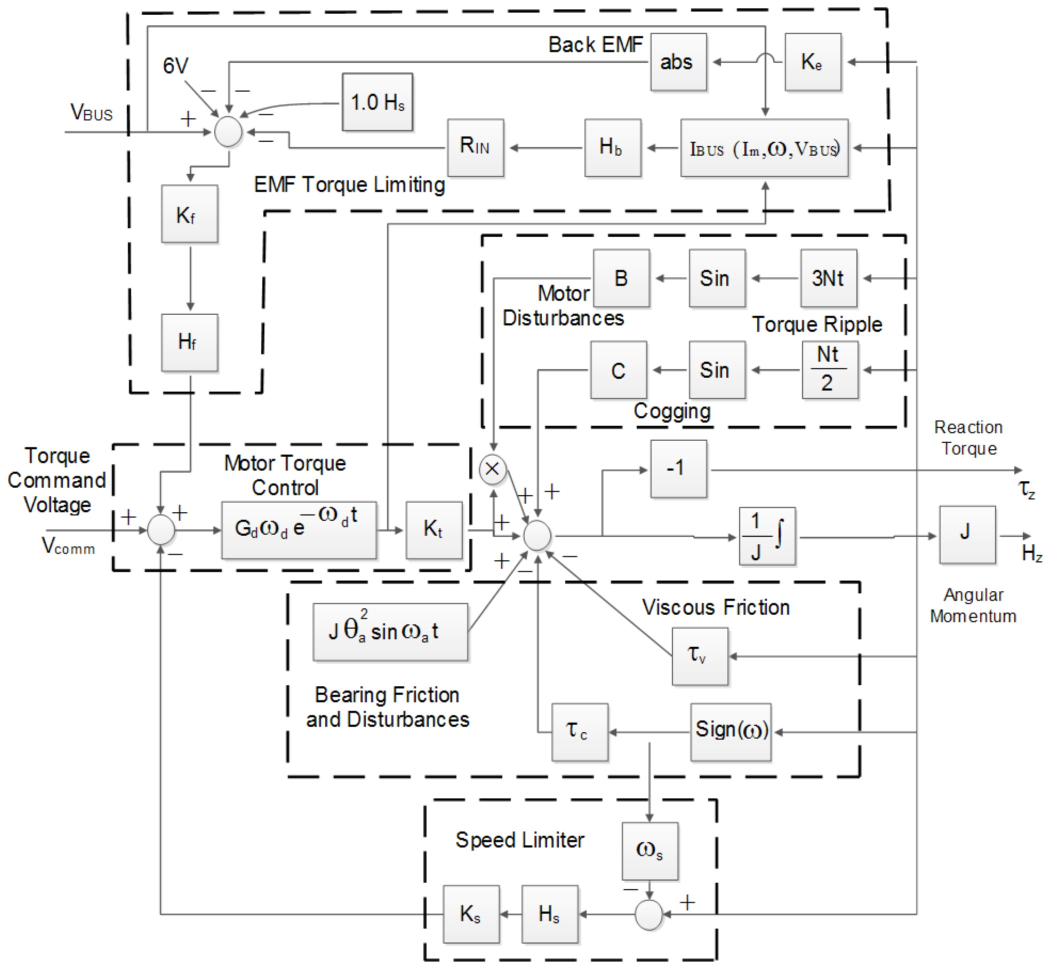

Figure 2.2 Reaction Wheel Model [59]

The expression for Coulomb friction is,

$$T_{coul} = \tau_c sign(\omega) \tag{2.3}$$

where τ_c is the coefficient of Coulomb friction with units $N \cdot m$, and the *sign* function can be characterized as shown below.

$$sign(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases} \tag{2.4}$$

Viscous friction varies depending on the type of the bearing lubricant and its temperature as well as the speed of the rotor. A higher bearing lubricant temperature will create less friction in the bearing while different lubricant materials will have varying viscosities. Viscous friction can be expressed as in Eq. (2.5),

$$T_{visc} = \tau_v \omega \qquad (2.5)$$

where τ_v is the coefficient of viscous friction with units $N.m/rad/s$. The motor torque block in Figure 2.2 consists of scaling the motor current by the torque constant (K_t, with SI units $N.m/A$) which is equal to the BEMF constant when expressed in SI units. For the remainder of this thesis the terms 'BEMF constant' and 'torque constant' are used interchangeably.

It is important to discuss the high degree of nonlinearity in this RW model. In particular, attention must be paid to the heavyside, absolute value, and signum functions in the model. These functions all lead to discontinuities that must be approximated by appropriate analytical models. These models take the form of rational exponential functions. For the numerical representation of this model refer to Ref. [18].

3. Kalman Filters

The Kalman Filter (KF) is a model-based observer that produces estimates based on the stochastic properties of a system. This feature of the KF makes it robust to measurement and process noise, hence, make it practical to implement. The most common application for KFs is in fault diagnosis and identification (FDI) which has been studied extensively in the literature. Three of the common KF variants applied to the FDI problem are the EKF, AKF, UKF, and adaptive unscented Kalman filter (AUKF).The EKF uses linearization to estimate the true mean and covariance of the random variable while the UKF uses sigma points or particles. These sigma points when spread through a non-linear system, capture the posterior mean and covariance accurate to the third order, whereas the standard EKF is only accurate to the first order [62]. This means that the UKF is a better option for highly non-linear systems but has no advantage for weakly non-linear systems. Whether the EKF or UKF structures are used the adaptive mechanism, which could be used in both, remains the same. These filters can be implemented for either state/parameter estimation, or joint state and parameter estimation. State estimation uses the standard KF equations without any modifications and hence, is the most straight forward approach to Kalman filtering. The goal in this approach is to estimate the system states based on the mathematical model of the system. The FDI approach could be as simple as just comparing the measured system states from the sensors and with the predicted model states, if the residuals exceed a threshold then a fault has been detected. On the other hand, isolation and identification are not as direct and generally require a good choice of residuals in addition to post-processing of the raw data. Parameter estimation, however, is a form of system identification as it implicates estimating the physical parameters of a system. In order to accommodate the parameters as the state-vector for this purpose, modifications need to be made to the KF equations. The FDI problems are then directly resolved when the parameters of a system are estimated; the reason for that is that the change in parameter(s) identifies where the faults have occurred and the level of severity for each. Having said that, the parameter estimation approach is better suited for FDI problems than the state-estimation is.

3.1 Unscented Kalman Filter

The Robotics Research Group (RRG) in Oxford UK proposed a "New Filter" in 1994 which was named the UKF. Then, in 1997 the first paper was published describing the UKF as a new extension of the KF to nonlinear systems . The UKF is a variant of the KF with the capability of estimating the mean and covariance of a random variable to the third order while the Extended Kalman Filter (EKF) only approximates them to the first order. As a result, higher order terms in the dynamics are not ignored in UKF. This filter is built on the belief that "it is easier to approximate a probability distribution than it is to approximate an arbitrary nonlinear function." [63]. Simply put, it means that the linear approximations are not used to approximate non-linear functions in here; instead, the statistical moment of the state is approximated. The UKF is, in fact, a form of particle filter applied to a random variable with Gaussian distribution. Generally particle filters can be applied to systems with sampling densities that are non-Guassian [64] where the posterior distribution of the state is approximated using a large number of "well chosen" *particles* or *sigma points* that changes randomly in time according to the model dynamics and system measurements [65]. Therefore the UKF is similar to the Monte Carlo simulation except for the part that the points are chosen deterministically in the UKF.

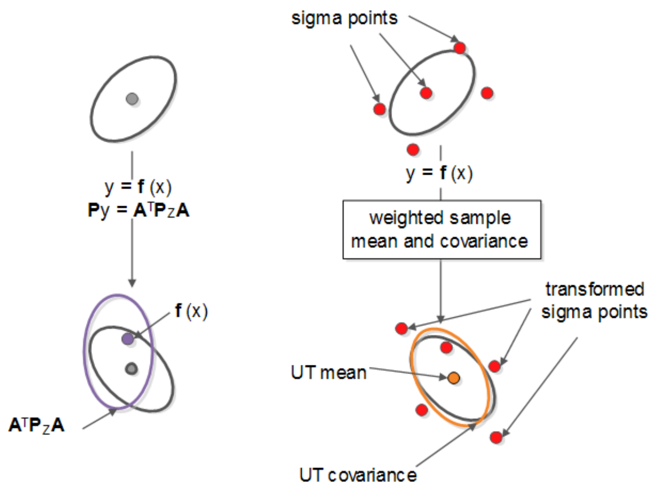

Figure 3.1 Linearization (on the left) vs. Unscented Transformation (on the right)

Figure 3.1 shows a visual explanation of the unscented transform versus linearized transform. Variable $\mathbf{y}$ is obtained by a propagating a random variable $\mathbf{x}$ through a non-linear function f. The goal is to estimate the mean $\bar{\mathbf{y}}$ and covariance $\mathbf{P}_{yy}$ of $\mathbf{y}$ as accurate as possible. As discussed earlier, because EKF linearization is only accurate to the first order, the statistical properties of the output vector can not be captured accurately. The unscented transform, on the other hand, generates a cloud of sigma points which has a covariance and mean closer to the real values. There are three steps to the unscented transformation process. First, given the n-state random variable $\mathbf{x}$ a set of $2n$ sigma points are generated around $\bar{\mathbf{x}}$ along with a set of $2n+1$ weights (one for $\bar{\mathbf{x}}$). Then the $2n$ points are fed to the non-linear output function h to obtain $\mathbf{y}$. And eventually the mean $\bar{\mathbf{y}}$ and covariance $\mathbf{P}_{yy}$ of $\mathbf{y}$ are calculated based on the distribution of these "particles" or sigma points and their weights. The UKF formulation is as follows [66]:

1) ***Compute weights***

$$\mathbf{W}_0 = \frac{\kappa}{n+\kappa} \tag{3.1}$$

$$\mathbf{W}_i = \frac{1}{2(n+\kappa)} \tag{3.2}$$

2) ***Establish symmetric sigma points about the state estimate***

$$\hat{\boldsymbol{\chi}}_0 = \hat{\mathbf{x}}_k^- \tag{3.3}$$

$$\hat{\boldsymbol{\chi}}_i = \hat{\mathbf{x}}_k^- + \sqrt{(n+\kappa)\mathbf{P}_i^-}, \qquad \forall i = 1,2,...,n \tag{3.4}$$

$$\hat{\boldsymbol{\chi}}_i = \hat{\mathbf{x}}_k^- - \sqrt{(n+\kappa)\mathbf{P}_i^-} \qquad \forall i = n+1,...,2n \tag{3.5}$$

3) ***Instantiate sigma points through process model***

$$\tilde{\boldsymbol{\chi}}_i = f(\hat{\boldsymbol{\chi}}_i) \tag{3.6}$$

4) ***Predict mean and covariance of states***

$$\bar{\mathbf{x}} = \sum_{i=0}^{2n} W_i \tilde{\boldsymbol{\chi}}_i \tag{3.7}$$

$$\mathbf{P}_{xx} = \sum_{i=0}^{2n} W_i [\tilde{\boldsymbol{\chi}}_i - \bar{\mathbf{x}}][\tilde{\boldsymbol{\chi}}_i - \bar{\mathbf{x}}]^T + \mathbf{Q} \tag{3.8}$$

5) ***Instantiate sigma points through measurement model***

$$\mathbf{Y}_i = h(\tilde{\boldsymbol{\chi}}_i) \tag{3.9}$$

6) ***Predict mean and covariance of measurements***

$$\bar{\mathbf{y}} = \sum_{i=0}^{2n} W_i \mathbf{Y}_i \tag{3.10}$$

$$\mathbf{P}_{yy} = \sum_{i=0}^{2n} W_i [\mathbf{Y}_i - \bar{\mathbf{y}}][\mathbf{Y}_i - \bar{\mathbf{y}}]^T + \mathbf{R} \tag{3.11}$$

7) ***Predict cross covariance***

$$\mathbf{P}_{xy} = \sum_{i=0}^{2n} W_i \left[\tilde{\boldsymbol{\chi}}_i - \bar{\mathbf{x}}\right]\left[\mathbf{Y}_i - \bar{\mathbf{y}}\right]^T \tag{3.12}$$

8) *Gain calculation and updates*

$$\mathbf{K} = \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1} \tag{3.13}$$

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + \mathbf{K}\left(\mathbf{y} - \bar{\mathbf{y}}\right) \tag{3.14}$$

$$\mathbf{P}^{+} = \mathbf{P}_{xx} - \mathbf{K}\mathbf{P}_{yy}\mathbf{K}^T \tag{3.15}$$

If $\mathbf{x}$ is assumed to be Gaussian then κ should be selected such that $n+\kappa=3$ [17], for different distributions another value may be more appropriate. In Eqs. (3.4) and (3.5) the $\sqrt{(n+\kappa)\mathbf{P}_i^-}$ terms represent the scaled i^{th} rows/columns of the square root-factor of $\mathbf{P}^-$. As it can be seen from the fomulation, there is no linearization in the filter, and instead sigma points are fed to the process and measurement models after which the state and measurement statistics are estimated. Figure 3.2 shows the flow of the algorithm.

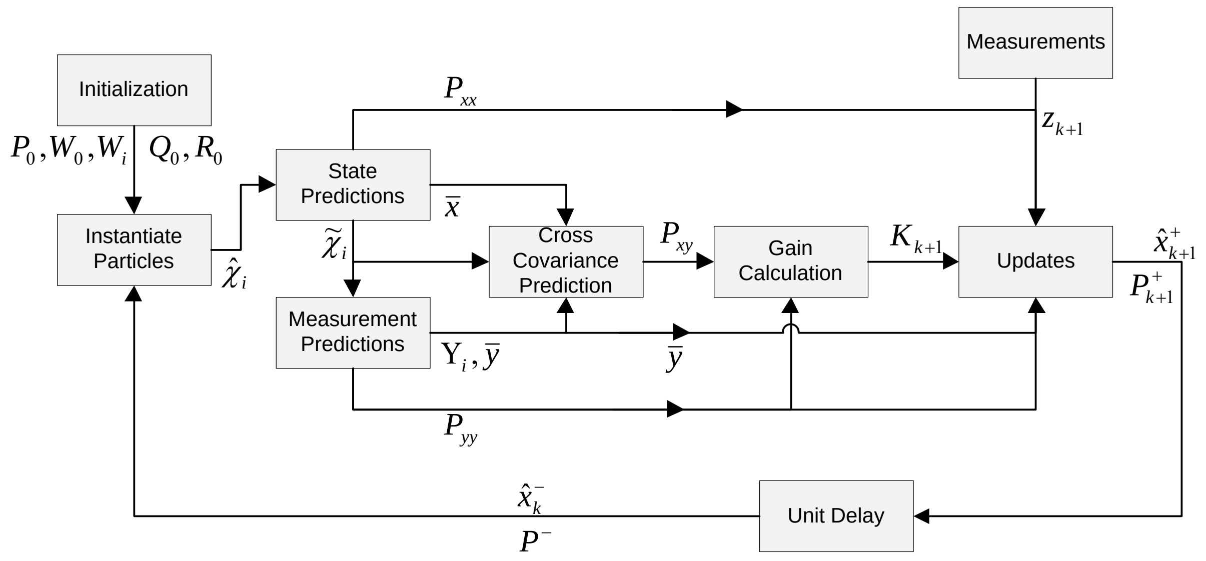

Figure 3.2 UKF Signal Flowchart [17]

The UKF has been successfully applied to FDI problems to resolve some of the issues associated with linearization in the EKF that could lead to false alarms. An extensive review on is done in Chapter 4 of Ref. [17].

3.2 Parameter Estimation with UKF

It was also indicated in the previous section that if the KF is formulated to perform parameter estimation, identification could be done in the filter with the least post-processing. Principally a fault is typically the result of a change in system parameters; therefore, direct estimation of these parameters can provide enoigh information on the type, severity, and location of the fault. The challege here is that the computational requirement for this method could be large once there are many "fault parameters" in a system, however, in reality only those parameters reflecting the most common or critical fault types of a system need to be to estimated. This technique also has the advantage of producing as a byproduct the mathematical model for the system in question which can be used to generate residuals; hence, state-estimation is not always necessary. An extensive literature review on the applications for this method is available in Chapter 4 of Ref. [17].

Thae fact that parameters are usually constant in a given mathematical model implies that their time-evolution can be refer to as $\dot{\mathbf{x}}_p = 0$. But in reality their estimates are actually time-varying. This leads to the conclusion that the state-prediction stages of the UKF must be modified to account for these aspects. Logically a common question arises as to "how the parameter estimates are varied if their time-evolution is zero?". The answer is that the evolution of these parameters is caused by the stochastic properties of the system. Poorly chosen $\mathbf{Q}$ and $\mathbf{R}$ matrices, hence, will lead to biases in the estimates and can even result in instability [67]. This is the main focus of this thesis as how to choose parameters for the filters systematically, Kalman filters specifically in this case, so that the estimations are accurate enough with the least residul in the system.

The formulation for parameter estimation with the UKF is as follow [68].

1) ***Compute weights***

$$\mathbf{W}_0 = \frac{\kappa}{n+\kappa} \tag{3.16}$$

$$\mathbf{W}_i = \frac{1}{2(n+\kappa)} \tag{3.17}$$

2) ***Establish symmetric sigma points about the state estimate***

$$\hat{\boldsymbol{\chi}}_0 = \hat{\mathbf{x}}^-_{p(k+1)} \tag{3.18}$$

$$\hat{\boldsymbol{\chi}}_i = \hat{\mathbf{x}}^-_{p(k+1)} + \sqrt{(n+\kappa)\mathbf{P}^-_{(k+1)i}}, \qquad \forall i = 1,2,...,n \tag{3.19}$$

$$\hat{\boldsymbol{\chi}}_i = \hat{\mathbf{x}}^-_{p(k+1)} - \sqrt{(n+\kappa)\mathbf{P}^-_{(k+1)i}} \qquad \forall i = n+1,...,2n \tag{3.20}$$

3)*Predict mean and covariance of states*

$$\mathbf{P}^-_{k+1} = \mathbf{P}^-_k + \mathbf{Q}_{k+1} \tag{3.21}$$

$$\mathbf{x}^-_{p(k+1)} = \mathbf{x}^+_{pk} \tag{3.22}$$

4) *Instantiate sigma points through measurement model*

$$\mathbf{Y}_i = g(\hat{\boldsymbol{\chi}}_i) \tag{3.23}$$

5) *Predict mean and covariance of measurements*

$$\overline{\mathbf{y}} = \sum_{i=0}^{2n} W_i \mathbf{Y}_i \tag{3.24}$$

$$\mathbf{P}_{yy} = \sum_{i=0}^{2n} W_i [\mathbf{Y}_i - \overline{\mathbf{y}}][\mathbf{Y}_i - \overline{\mathbf{y}}]^T + \mathbf{R}_{k+1} \tag{3.25}$$

6) *Predict cross covariance*

$$\mathbf{P}_{xy} = \sum_{i=0}^{2n} W_i \left[\hat{\boldsymbol{\chi}}_i - \hat{\mathbf{x}}^-_{p(k+1)}\right][\mathbf{Y}_i - \overline{\mathbf{y}}]^T \tag{3.26}$$

7) *Gain calculation and updates*

$$\mathbf{K} = \mathbf{P}_{xy}\mathbf{P}^{-1}_{yy} \tag{3.27}$$

$$\hat{\mathbf{x}}_p = \mathbf{x}^-_{p(k+1)} + \mathbf{K}(\mathbf{y} - \overline{\mathbf{y}}) \tag{3.28}$$

$$\mathbf{P}^+_{k+1} = \mathbf{P}^-_{k+1} - \mathbf{K}\mathbf{P}_{yy}\mathbf{K}^T \tag{3.29}$$

As it can be noticed, the main difference in this implementation is in steps 3 and 4. Since there is no dynamics information available for parameters, the sigma points from step 2 are not propagated; instead, they are used in subsequent steps as before. The "predicted" mean and covariance of the states for the state-estimation UKF, $\overline{\mathbf{x}}$ and $\mathbf{P}_{xx}$ respectively, in Eqs. (3.7) and (3.8) are replaced throughout the remaining steps by the predicted mean and covariance $\mathbf{x}^-_{p(k+1)}$ and $\mathbf{P}^-_{k+1}$ of the states in step 3 here. The differences between the flow of this algorithm an that of the state-estimation UKF lie in the state prediction calculations.

The UKF does not require a measurement matrix because of the unscented transform.

Therefore, there is no uncertainty for the residual generation form. The function h in step 5 of this algorithm represents the output equation of the system in terms of its parameters.

When using KFs for parameter estimation, $\boldsymbol{Q}_{k+1} / \boldsymbol{R}_{K+1}$ ratio becomes an important factor in the overall performance of the filter because ultimately parameter changes in a physical system represent a change in the system model. When running the filter this should be reflected as an increase in process noise covariance matrix entries or the bandwidth of the filter in general. otherwise the algorithm will only be able to perform the identification of parameters once. This happens due to the fact that when the Kalman gain reaches the steady-state period and all the parameters are estimated correctly, the filter bandwidth is usually near-zero; hence, if any of these parameters change then considering the bandwidth of the filter being fixed, it will not be able to track the changes. AKFs provide the capability for the filters to make the identification possible in the case of parameter uncertainty by adapting filter bandwidth based on residuals of measured changes in the system behaviour versus the modeled behavior.

3.3 Adaptive Kalman Filtering

Adaptive Kalman filtering has been around since the late 1960's for online estimation of measurement and/or process noise characteristics [69, 70]. At the early stages of KF research it was understood that in practice stochastic noise properties of a system are not always known. A mechanism was needed that could recursively compute the real measurement and process noise statistics online so that variations in system uncertainty could be tracked. These methods have been documented in [71], three of the most common are; (1) the *Bayesian approach*, (2) the *maximum likelihood approach*, and (3) *innovation/residual-based approach*. An extensive review on these methods is done in Ref. [17].

Among these methods, to relax the requirements on a priori information of the measurement and process noise structures, innovation/residual-based approaches can be considered. Most current adaptive algorithms use some form of processing of the innovation/residual sequence to obtain better estimates of system statistics, better tracking, and faster convergence. Algorithms such as adaptive fading factor (AFF) and covariance matching (CM) fall into this category of adaptive filters. AFF algorithms generally introduce a scale-factor α to the error covariance prediction and/or gain calculations. In Reference [51] a scaling factor

α is defined as a function of estimated and theoretical innovation covariances $\hat{\mathbf{C}}_v$ and $\mathbf{C}_v$ respectively shown in the equations below,

$$\alpha = max\left\{1, \frac{1}{N} trace\left(\hat{\mathbf{C}}_v \mathbf{C}_v^{-1}\right)\right\} \tag{3.30}$$

$$\hat{\mathbf{C}}_v = \frac{1}{N} \sum_{j=k-N+1}^{k} \Delta\hat{\mathbf{x}}_k \Delta\hat{\mathbf{x}}_k^T \tag{3.31}$$

$$\mathbf{C}_v = \mathbf{R}_{k+1} + H_{k+1}\mathbf{P}_{k+1}^{-}H_{k+1}^T \tag{3.32}$$

where $'trace'$ is the trace operator. This has the affect of modifying the scale-factor when estimated variances based on innovations become larger. In other words when changes in system parameters occur the mechanism causes changes in the bandwidth of the filter. This is done by multiplying the gain of the standard KF equations by $1/\alpha$. Another form of adapting the scale-factor is proposed in [72] where the magnitude of the deviation of the innovation vector from zero is used as an input to fuzzy rules, which then output a scale-factor representing the degree of confidence that divergence is occurring. Research presented in [73] demonstrates the AFF method using both the residual and innovation sequences and concludes that a fading factor expressed by the innovation sequence is superior to one expressed by the residual sequence. The CM approach is a method of making residuals and innovations consistent with their theoretical covariances. Usually either the $\mathbf{R}$ matrix is held constant while the $\mathbf{Q}$ matrix is adapted or vice-versa. In the former case the estimated innovation covariance $\hat{\mathbf{C}}_v$ is used to adapt the $\mathbf{Q}$ matrix until it matches the theoretical covariance [74]. As innovations become larger in the face of system faults the estimated covariance increases thereby increasing $\mathbf{Q}$ and therefore the Kalman gain. This method can be subject to abrupt changes in $\mathbf{Q}$ in which case a running average window can be used to smooth out the estimate [75]. If $\mathbf{R}$ is to be estimated while $\mathbf{Q}$ is held constant, $\mathbf{R}$ is adapted based on the estimated residual sequence covariance $\hat{\mathbf{C}}_r$ until the covariances match. In reference [76] this method is used along with fuzzy rules to identify the amount and direction of change that should occur in the measurement noise matrix. These methods are sub-optimal as they involve approximations to the true statistics and in some cases convergence is uncertain. However they are more robust and responsive than the Bayesian and MLA methods because information from the residual and/or innovation sequences provide close approximations to the actual variances.

Traditional covariance matching techniques estimate either the measurement or process noise matrices while the other is assumed constant, adaptive sequential estimation is a similar technique to that of covariance matching except that both measurement and process noise statistics are estimated simultaneously online. Myers and Tapley [77] were one of the first to propose such a method. First they define an unbiased estimator for residual $\mathbf{r}_{k+1}$ as the following sample mean

$$\bar{\mathbf{r}}_{k+1} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{r}_j \tag{3.33}$$

Next an estimate of the covariance of $\bar{\mathbf{r}}_{k+1}$ is calculated along with its expected value.

$$\bar{\mathbf{C}}_r = \frac{1}{N-1}\sum_{j=1}^{N}\left(\mathbf{r}_j - \bar{\mathbf{r}}_{k+1}\right)\left(\mathbf{r}_j - \bar{\mathbf{r}}_{k+1}\right)^T \tag{3.34}$$

$$E\left[\bar{\mathbf{C}}_r\right] = \frac{1}{N}\sum_{j=1}^{N}H_j\mathbf{P}_{k+1}^{-}H_j^T + \mathbf{R} \tag{3.35}$$

The resulting unbiased estimate is shown below.

$$\hat{\mathbf{R}} = \frac{1}{N-1}\sum_{j=1}^{N}\left[\left(\mathbf{r}_j - \bar{\mathbf{r}}_{k+1}\right)\left(\mathbf{r}_j - \bar{\mathbf{r}}_{k+1}\right)^T - \left(\frac{N-1}{N}\right)H_j\mathbf{P}_{k+1}^{-}H_j^T\right] \tag{3.36}$$

Process noise can be estimated in a similar fashion except using the innovation sequence,

$$\hat{\mathbf{Q}} = \frac{1}{N-1}\sum_{j=1}^{N}\left[\left(\Delta\hat{\mathbf{x}}_{kj} - \Delta\bar{\hat{\mathbf{x}}}\right)\left(\Delta\hat{\mathbf{x}}_{kj} - \Delta\bar{\hat{\mathbf{x}}}\right)^T - \left(\frac{N-1}{N}\right)\left(\mathbf{\Phi}_{k+1}\mathbf{P}_{k+1}^{-}\mathbf{\Phi}_{k+1}^T - \mathbf{P}_{k+1}^{+}\right)\right] \tag{3.37}$$

where $\Delta\bar{\hat{\mathbf{x}}}$ is the mean of the innovation. In this algorithm the difference between the innovation/residual vectors and their respective running average means is used to obtain covariance estimates. The goal being to obtain process and measurement noise estimates for covariance matching. Absolute values of diagonal entries of $\hat{\mathbf{Q}}$ and $\hat{\mathbf{R}}$ must be taken in order to guarantee the positive definiteness of these matrices. An extensive review on different sapproaches to AKF is presneted in Chapter 4 of Ref. [17] and at the end, the author proposed a new approach for the AFF calculation which is furhter explained in the next section.

3.4 FDI by Parameter Estimation with Adaptive Kalman Filters

To approach the problem of FDI for ACS hardware such as RWs, AUKF is presented for parameter estimation assuming full state measurement. A primary goal in the design, as claimed by the author in Ref. [17], was to limit the computational necessities of the algorithm to help implement it in the ACS module. This means that only parameter estimation is considered to limit the prediction and update equation's computational requirements. A joint n-state and p-parameter estimator would result in an augmented state-vector $\mathbf{x}_a$ with dimension $n+p$. Assuming m measurements are available, dimensions of the error covariance matrix would increase to $(n+p)\times(n+p)$, while those of the gain matrix would be $(n+p)\times m$. For $n=2$, $m=2$, and $p=2$ the covariance matrix for parameter only estimation has four entries while in the joint case this number increases by a factor of 4. Similar computational savings are observed in the gain matrix to a lesser degree, however on macroscopic scales of time these computational savings quickly add up. State-propagation is performed by running a model online whose outputs are conditioned by the estimated parameters. Although this entails excess computational requirements, they are not realized in the KF equations, and the net savings is still considerable since only one set of equations needs to be calculated.

In order to better understand the reasons behind choosing KF for the purpose of FDI can be found in Chapter 4 of Ref. [17] and here in order to avoid unnecessary repetiotion, the reader is referred to the main reference and only the importan formulation and explanations are included.

The following adaptive mechanism is applied to the EKF formulation

$$\hat{\mathbf{C}}_r = \frac{1}{N}\sum_{j=k-N+1}^{k} \mathbf{r}_k \mathbf{r}_k^T \tag{3.38}$$

$$\hat{\mathbf{R}}_{k+1} = \hat{\mathbf{C}}_r + H_{k+1}\mathbf{P}_{k+1}^{+}H_{k+1}^T \tag{3.39}$$

$$\hat{\mathbf{Q}}_{k+1} = \mathbf{K}_{k+1}\hat{\mathbf{C}}_r\mathbf{K}_{k+1}^T \tag{3.40}$$

The moving window average of the matrix in Eq. (3.38) is updated at each time-step. In practice the elements of the moving window are stored as an array.

At each iteration the oldest element in the array is shifted out while the newest residual vector is shifted in, then the measurement and process noise estimates are calculated. Selection of the window size depends on the application, Ref. [52] provides criteria for window length selection to avoid divergence and/or instability; (1) *A window size smaller than the number of measurements when adapting R.* (2) *A window size smaller than the number of filter states when adapting Q.* (3) *A window size smaller than the sum of update measurements and filter states when adapting both Q and R.*

In these three cases divergence occurs because there are less equations than unknown parameters, resulting in an under-determined system. Following the above criteria destabilization of the filter is averted, however biased estimates may result for small sample sizes. For unbiased estimates a larger window length is preferred, however a window length that is too large will not allow the filter to correctly track high-frequency changes in the system states. Consequently the lower bound of the window length is selected based on the number of filter states and measurements, while the upper bound is selected depending on the dynamics of the system. For implementation in the UKF a modification must be made to Eq. (3.39) because the measurement matrix is no longer available. Estimated measurement covariance is calculated in the UKF using Eq. (3.25), consequently the measurement noise covariance matrix should be calculated as [78],

$$\hat{\mathbf{R}}_{k+1} = \hat{\mathbf{C}}_r + \mathbf{P}^*_{yy} \tag{3.41}$$

$$\mathbf{P}^*_{yy} = \sum_{i=0}^{2n} W_i \left[\mathbf{Y}_i - \bar{\mathbf{y}}\right]\left[\mathbf{Y}_i - \bar{\mathbf{y}}\right]^T \tag{3.42}$$

where $\hat{\mathbf{C}}_r$ is the same as in Eq. (3.38). The signal flow of the resulting AUKF algorithm is shown in the block diagram below.

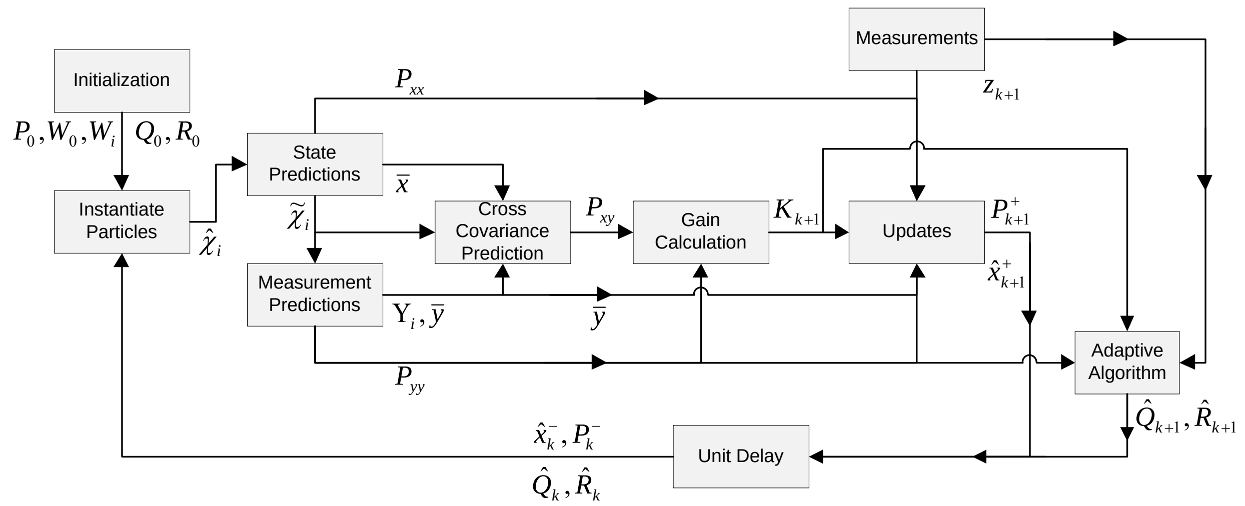

Figure 3.3 AUKF Signal Flowchart [17]

In this case the prediction stage of the UKF is

$$\tilde{\boldsymbol{\chi}}_i = \hat{\boldsymbol{\chi}}_i \tag{3.43}$$

$$\mathbf{P}_{xx} = \mathbf{P}_k^- + \mathbf{Q}_{k+1} \tag{3.44}$$

In addition to adapting the $\mathbf{Q}$ and $\mathbf{R}$ matrices, a fading factor is applied to the error covariance matrix in the prediction stage for the UKF as in Ref. [17]

$$\mathbf{P}_{xx} = \mathbf{P}_k^- \varepsilon + \mathbf{Q}_{k+1} \tag{3.45}$$

As it is further explained in Ref. [17], If $\varepsilon = 1$ then the standard KF prediction occurs, if $\varepsilon > 1$ the filter will weight the data exponentially so that the effect of current data is emphasized and information from older measurements is discounted, hence the name 'fading factor' or 'fading memory'. In the standard KF algorithm estimates depend highly upon past data which can lead to divergence of the estimates even in the face of new measurements. An FDI algorithm should consider current data more heavily so that estimates can track the current state of a system. In essence the fading factor limits how small the error covariances can get by artificially inflating the value of the predicted error covariance matrix thus introducing more uncertainty into the system. In Ref. [79] it is shown that larger values of ε give the filter a larger bandwidth with the opposite happening for a smaller value. Typically the fading factor is in the range $1 < \varepsilon < 1.01$ however the appropriate choice depends on the particular application. If it is close to or larger than the upper bound then instability can ensue, while if it is close to or lower than the lower bound there will be no effect.

The algorithm explained in Chapter 4 of Ref. [17] incorporates two adaptive mechanisms; one to ensure that changes in system parameters are reflected as increased modeling errors, and the other to limit the memory of the filter so that it pays more attention to current data. Ultimately the modified algorithm attempts to adapt the bandwidth of the filter based on a moving window average of residuals while making sure that the bandwidth does not get small enough so that the filter ignores new data. To make the algorithm more robust and accurate an adaptive fading factor is used. The primary goal is to force the filter to consider new measurements more heavily when faults occur and less heavily when no further faults are detected. Although the noise covariance estimations perform the function of adapting the filter bandwidth, eventually the filter will converge to very small gains thus making the bandwidth very small. In this case even if a fault occurs and the $\mathbf{R}$ matrix becomes large, the $\mathbf{Q}$ matrix will be much smaller because it is a function of the square of the gain matrix. Consequently the filter will not track correctly. A forgetting factor can mitigate this effect be forcing the filter to forget older data and become more sensitive to newer data. However a constant forgetting factor can have adverse effects when convergence has been achieved, where it would be desired to make the filter less sensitive to new data so that the estimate holds even in the face of disturbances. An adaptive forgetting factor could force the filter to ignore new data when estimates converge and consider new data more heavily in the opposite case. This adaptation is based on the magnitude of the $\mathbf{R}$ matrix. Large diagonal entries of the measurement noise covariance matrix $\mathbf{R}$ result from large residuals as per Eq. (3.38). Because a running average of the residuals is used a brief disturbance will not be detected, however a disturbance that persists over a longer period of time will be reflected in the residual average. Thus adapting the fading factor based on the magnitude of the $\mathbf{R}$ matrix would ensure that when the residuals become larger the filter bandwidth opens up, with the opposite happening when residuals become smaller. The adaptation is as follows [17],

$$\varepsilon = 1 + \frac{trace\left(\mathbf{R}_k\right)}{\zeta} \tag{3.46}$$

This formulation guarantees that the adaptive factor will increase for larger values of $\mathbf{R}_k$. The $trace$ operation consists of the sum of the diagonal elements of a matrix, which indicates the size of the residual error. Here ζ is one of the parameters in the optimization for the RW FDI.

4. Particle Swarm Optimization

In mathematics, computer science, or management science, mathematical optimization, also known as optimization or mathematical programming, refers to the selection of the best solution from a set of available alternatives.

In the simplest case, an optimization problem is based on either maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function. The generalization of optimization theory and techniques to other formulations comprises a large area of applied mathematics. More generally, optimization includes finding "best available" values of an objective function given a defined domain.

In this Chapter, an introduction to optimization and different algorithms used in this area is provided. In general, optimization is commonly used in engineering and social based problems and in all of the optimization problems there are three main lemmas:

1. Considering all the independent variables of the problem
2. Forming the objective function based on these variables
3. Fixing constraints of the problem

Hence, an optimization problem is usually formed in the standard form of

$$\begin{aligned} &Max/Min: F(x) \\ &g_i(x) <= 0 \quad i = 1,2,3,\ldots,p \ \forall \\ &H_j(x) = 0 \qquad j = 1,2,3,\ldots,q \\ &\chi_{k\min} \leq \chi_k \leq \chi_{k\max} \end{aligned} \tag{4.1}$$

where x is the design or parameter vector, which represents the parameters that are optimized by the algorithm, $F(x)$ is the objective function and $g_i(x)$, $H_i(x)$ are the inequality and equality constraints of the problem, respectively.

4.1 Brief History

Major subfields of the optimization include: *Convex programming* which studies the case when the objective function is convex (minimization) or concave (maximization) and the constraint set is convex. This can be viewed as a particular case of nonlinear programming or as generalization of linear or convex quadratic programming. *Linear programming (LP)* is another subfield, a type of convex programming, which studies the case where the objective function f is linear and the set of constraints is detailed using only linear equalities and inequalities. In this subfield, such a set is known as a polyhedron or a polytope if it is bounded. There is another major subfield called *Second order cone programming (SOCP)* which is a convex program, and includes certain types of quadratic programs. *Semi definite programming (SDP)* on the other hand, is a subfield of convex optimization where the underlying variables are semi definite matrices. It subfield is somehow a generalization of linear and convex quadratic programming. *Conic programming* is, in practice, a general form of convex programming. LP, SOCP and SDP can all fall into the category of conic programs with the appropriate type of cones. *Geometric programming* is a technique in which the objective and inequality constraints are expressed as posynomials and equality constraints as monomials can be transformed into a convex program. *Integer programming* is another subfield, which studies linear programs wherein some or all variables are constrained to take on only integer values. This is important to notice that this subfield is not same as convex, and in general is much more difficult than the regular linear programming. Another subfield is the *Quadratic programming*, which allows the objective function to have quadratic terms, while the feasible set must be specified with linear equalities and inequalities. Interestingly, for specific forms of the quadratic term, this is a type of convex programming. *Fractional programming* studies optimization of ratios of two nonlinear functions. The special class of concave fractional programs can be transformed to a convex optimization problem. The general case in which the objective function or the constraints or both contain nonlinear parts is studied in *Nonlinear programming*. *Stochastic programming* is a subfield, which studies the case wherein some of the constraints or parameters depend on random variables. However, *Robust programming* is, like stochastic programming, an attempt to capture uncertainty in the data underlying the optimization problem but with a slight difference that this is not done by random variables and the problem is solved by taking inaccuracies in the input data into account. *Combinatorial optimization* is concerned with problems where the set of feasible solutions is mainly discrete or there is a possibility for it to reduce to a discrete one. *Infinite-dimensional optimization* studies the case where the set of feasible solutions is a subset of an infinite-dimensional space, such as a space of functions. *Heuristics and metaheuristics* make few or no assumptions about the problem being optimized. Hence, there is no need for derivatives in this subfield. Usually, heuristics do not guarantee that any optimal solution would be found. On the other hand, heuristics are used to find approximate solutions for many complicated optimization problems. More explanations on this subfield are given later in this Chapter. *Constraint satisfaction* subfield studies the case in which the objective function f is constant. This subfield is particularly useful in artificial intelligence in the automated reasoning. When there is a need for at least one constraint to be satisfied but not all *Disjunctive programming* is used. It is of particular use in scheduling.

In a number of subfields, the techniques are aimed mainly for optimization in dynamic contexts which means decision making over time: *Calculus of variations* is one of the which seeks to optimize an objective defined over many points in time, by considering how the objective function behaves if there is a small change in the choice path. A generalization of the calculus of variations would be the *Optimal control theory*. *Dynamic programming*, on the other hand, studies the case where the optimization strategy is based on dividing the main into smaller sub-problems using the Bellman equation. *Mathematical programming with equilibrium constraints* is used when the constraints include variable inequalities or complementarities. In the following section, a brief review on different optimization algorithms is presented.

4.2 Particle Swarm Algorithm

The algorithm being used in this thesis is the Particle Swarm Optimization. PSO is a metaheuristic algorithm as it makes few or no assumptions about the problem being optimized and hence, can search very large spaces of candidate solutions. However, metaheuristics such as PSO do not guarantee an optimal solution is ever found [16]. More specifically, PSO does not use the derivative of any function for the problem being optimized, which means that it does not require the optimization problem to be differentiable as opposed to the classic optimization methods such as gradient descent and quasi-newton. Considering that, PSO can be used for optimization problems that are partially irregular, noisy, change over time, etc.

In computer science, particle swarm optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality known as “eliteness” or “fitness”. It optimizes a problem by having a population of candidate solutions, here known as particles, and moving these particles around in the search-space according to simple mathematical formulae acting on the particle's position and velocity. Each particle's movement is influenced by its local best-known position and is guided toward the best-known positions in the search-space, which are updated though time by the particle itself or other particles. This is expected to move the swarm toward the best solutions in the space being searched. The main advantage for this algorithm is its population, which makes the algorithm robust and free of being stuck in the local optimums as explained briefly in the previous section.

In terms of categorization, PSO falls into the category of evolutionary algorithms and the sub-filed of swarm intelligence based approaches, similar to algorithms such as Taboo Search and Ant Colony. Since its original development by Kennedy, Eberhart and Shi [60, 80] in 1995 PSO has mainly been applied to continuous-discrete heterogeneous strongly non-linear numerical optimization and it is thus used almost everywhere in the world. It was first intended for simulating social behaviour [81] as a stylized representation of the movement of organisms in a bird flock or fish school. Its convergence rate also makes it a preferred tool in dynamic optimization. The algorithm has been modified in many variations since then and being applied to many different optimization problems in different fields such as controller tuning [82, 83], trajectory design [84] and many other applications in the field of engineering and non-engineering problems. An extensive survey of PSO applications is made by Poli [85].

In Particle Swarm Optimization, an iterative procedure is followed to improve the results for the defined objective function by moving the particles in the search-space based on the reasoning as follow: Particles in this algorithm evaluate their “fitness” continuously and in each iteration, memorizing the best position that they have been to so far in their movement history. They also know the position of the best particle in the group. With these factors, particles in the swarm move in the n-dimensional space, foraging the solution. These particles have two outstanding characteristics:

1. Memory for storing the best position that they have been to.
2. Knowledge of the best particle in the swarm with the best position in the search-space.

Particles in the group are then in communication with each other and update these parameters in each iteration. They try to change their position and velocity while moving toward the best position based on the following information:

1. "global best" which is the best position of the whole group and is being updated in each iteration. Therefore, if the best position is changed after one iteration then the whole group would know what the position of the new "global best" is.
2. "local best" which is the best position of the particle in its movement history.

All the particles in the group try to move toward the "global best". The particles in the group search the area near the "global best" and do not search other areas of the search space. This phenomenon is called "convergence". If the inertia weight is chosen to be small, then all particles can reduce their speed so that when they reach the "global best", their speeds converge to zero. One way of getting out of an unpleasant "convergence" is to give the particles a new set of initial values after this unpleasant "convergence" occurs.

In general, the advantages of the PSO over GA algorithm are twofold: first is that in PSO particles use their history and the best of the group history to decide on the next move in the space. On the other hand, in GA transferring the knowledge of the current group to the next is thorough inheritance. This inheritance is affected by random procedures of cross-over and mutation and is not necessarily transferring a pure knowledge from one generation to the next; hence, this difference makes PSO faster and more reliable in comparison to GA algorithm.

Secondly, in GA particles in each iteration are subject to death and reborn and this puts a lot of computational burden on the system whereas in PSO particles only update their position and velocity and the population of the particles remains the same; hence, the computational burden would not be the problem anymore [86].

4.2.1 Algorithm

The original PSO formulae define each particle as a potential solution to the problem in the N-dimensional space. The position of ith particle is denoted as

$$X_i = (x_{i1}, x_{i2}, ..., x_{in}) \tag{4.2}$$

each particle also maintains a memory of its previous best position, stored in

$$P_i = (p_{i1}, p_{i2}, ..., p_{in}) \tag{4.3}$$

particles also move in the swarm with an individual velocity for each as below

$$V_i = (v_{i1}, v_{i2}, ..., v_{in}) \tag{4.4}$$

Each particle knows its best value so far (p_{best}) and its position. Moreover, each particle knows the best value in the swarm (g_{best}) among all p_{best}s. This information is analogy of knowledge of how the other particles around them have performed. Using that, each particle modifies its position using the 2 factors: 1) the distance between the current position and p_{best}, which as explained above, is the best value obtained in the history of the particle; and 2) the distance between the current position and g_{best}, which as explained above is the best value obtained in the whole swarm to this point. This modification is accomplished through the concept of velocity. The velocity of each agent or particle is altered using the following equation in Inertia Weight Approach (IWA)

$$v_i = w \cdot v_i + c_1 \cdot r_1 (P_i - X_i) + c_2 \cdot r_2 (P_g - X_i) \tag{4.5}$$

where, v_i is velocity of the particle, X_i is current position of the particle, w is the inertia factor which controls the influence of previous velocity on the new velocity, c_1 is a positive constant, called coefficient of the self-recognition component and determines the relative influence of the cognitive component, c_2 is a positive constant, called coefficient of the social component and determines the relative influence of the social component, P_i is p_{best} of particle i, P_g is g_{best} of the swarm and r_1, r_2 are random numbers used to maintain the diversity of the population, and are uniformly distributed in the interval (0,1).

Each particle decides where to move next using Eq. (4.5) which combines its own experience, which is the memory of its best past position and the experience of the most successful particle in the swarm. In the original PSO model, particles exploration the search-space within a range (−s, s) where s is any real number.

In this work, the inertia factor in Eq. (4.5) is set to 1.0 and remains constant throughout the whole optimization progression. The reason is that it is used to have a dynamic effect on the convergence of the swarm and by setting it to 1.0 the algorithm would have a uniform convergence rate during the optimization advancement and there is no need to further put time and effort to tweak w_{min} and w_{max} in Eq. (4.6). However, in Ref. [87] the authors suggest that the inertia factor can be calculated in each iteration using

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times iter \tag{4.6}$$

where w_{max} is the initial weight, w_{min} is the final weight, $iter_{max}$ is the maximum number of iterations and $iter$ is the current iteration number. Using the above equation, diversification characteristic is gradually decreased and a certain velocity, which gradually moves the current searching point close to p_{best} and g_{best} can be calculated but using it has the disadvantage that was mentioned earlier in this section.

The current position of each particle can then be modified by the means of:

$$X_i^+ = X_i^- + v_i \tag{4.7}$$

where X_i^+ is the new position, X_i^- is the old position and v_i is the particle's velocity calculated in Eq. (4.5). All swarm particles tend to move towards better positions; hence, the best position (i.e. optimum solution) will eventually be obtained through the combined effort of the whole population.

There are different termination criteria for the PSO algorithm [117]. One of the most common and widely used termination criteria is stopping the algorithm when the maximum number of iterations is reached and then the algorithm gives the "best solution found to this point" as the optimal solution for the problem. Figure 4.2 shows a simple flowchart of how the PSO algorithm works with this termination criterion. Another termination criterion is the difference between the last couple of solutions obtained in the process and if the difference is less than the tolerance value set in the algorithm, then the solution is set as the best solution found in the search-space ignoring the maximum number of iterations.

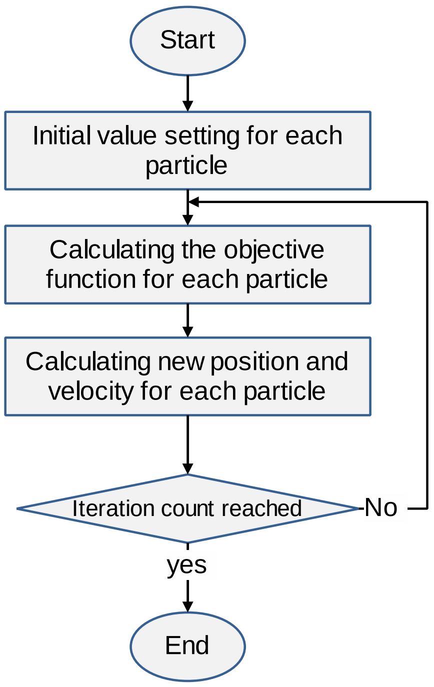
Start
Initial value setting for each particle
Calculating the objective function for each particle
Calculating new position and velocity for each particle
Iteration count reached
No
yes
End

Figure 4.1 PSO Algorithm Flowchart

4.2.2 Convergence criteria

In order to achieve convergence, there are some requirements to be satisfied. These requirements enforce the algorithm to output the results only when they are qualified for the solution. Ideally, there are two main criteria to be satisfied:

1. There is no change in the objective function value for a number of iterations.
2. There is no change in the particles' position for a number of iterations.

The number of iterations within which there should be no change in the values and positions is up to the designer but 5 to 10 is a reasonable number to start with. As mentioned before, the zero tolerance is the an ideal case and any small number could be used for the tolerance between the last few iterations the only difference would be the accuracy of the results and the choice depends on the severity, complexity and sensitivity of the problem in question. It also depends on the order of the values evaluated in the optimization because usually what is important is the percentage of the error and as long as it remains within a reasonable range. In this thesis, in the first 2 cases (Case 1 and Case 2) provided in Chapter 5, the convergence criteria is used with the tolerance of 1×10^{-6} but in Case 3, as explained in Chapter 5, the stopping criteria is just the number of iterations pre-set in the simulation. The reason is the lack of computing resources, especially the memory of the computing PC that the simulations were executed on, because all the information are written into the memory and read from it and when the number of iterations and particles are more than the computer can handle it, the simulation stops and the results are deleted.

In order to prevent this issue, it was suggested that instead of writing and reading from the memory, the information should be written and read from the hard disk drive of the operating system. This way the loss of data would be prevented and the computational limitation would be compensated for.

4.2.3 Example

In order to understand how the particles move toward the solution in the search-space, here an example is provided. The function of interest here is the Rastrigin function [88] defined as follow:

$$f\left(x,y\right)=20+\left(x^2-10\cos\left(\frac{\pi}{2}x\right)\right)+\left(y^2-10\cos\left(\frac{\pi}{2}y\right)\right) \tag{4.8}$$

The surface of this function is as shown in Figure 4.2. The objective here is to find the minimum of the function in the area shown. Selective iterations of the convergence and performance of the algorithm are shown in Figure 4.3. The optimization parameters in this example are given in Table 4.1. These parameters are consistent with Eq. (4.5). Having a look at the numbers in the table, it is evident that the algorithm is working fast while giving precise results as if the function was optimized using derivative-base algorithms. Number of iterations here is set to 50 but all the 80 particles in the swarm get to the final optimum solution after 30 iterations, which means that the algorithm is continuing for 20 more iterations without improving the results. This happens when the stopping/convergence criteria is set to reaching the maximum number of iterations and not any small change or difference between the last couple of best solutions acquired in the algorithm. The advantages and disadvantages to each of these rules for finishing the iterations is extensive discussed in the literature and in Ref. [89] it is more specifically focusing on a single-objective PSO stopping criteria. In the conclusions section of this reference, the author concludes, “It is not possible to determine one criterion that is best for all problems” and hence, the stopping criteria selection depends on the problem and the solver and there is no best solution for that in a generalized formulation.

Table 4.1 Optimization parameters for the example

Parameter	Value
$Run\,time\ [sec]$	0.3
$max\,Iterations$	50
$Particles$	80
w	1.0
c_1, c_2	1.0
$x, y\ range$	(-5, 5)
$final\ optimal\ at$	(0, 0)

Another observation is the number of particles; it is true that the more particles, the better the results but the tricky part in this statement is that there is no disclaimer on whether these better results could be achieved with fewer particles as well or not. Therefore, the challenge here and with almost all population-based optimization algorithms is how to choose these numbers and how to justify the choice. For the justification on the selection of w, c_1 and c_2 it is evident that these values are all set to 1.0 which means that they all have the same level of importance and their effect on the velocity evaluation function (Eq. (4.5)) is of equivalent significance. These values were set to 1.0 mainly because this is the easiest choice and the most commonly used value in the literature. The range for x, y variables was equally set to (-5 5) because from the function representation in Figure 4.2 the optimum point is known to be within this range for testing the algorithm with the function in Eq. (4.8).

By investigating the iterations and the trend, it can be seen that in the beginning the particles (shown by little squares with black borders in Figure 4.3) are scattered all over the search-space. As the time passes by and the iterations forge ahead, particles tend to converge toward the best particle of the swarm position and eventually this behaviour leads the swarm toward the best possible solution. This point is (0, 0) and it is noticeably observable in Figure 4.2 that this point is the global optimal of the function in Eq. (4.8).

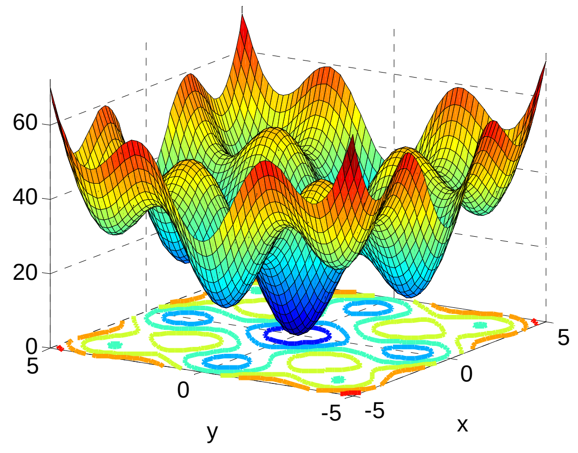

Figure 4.2 Surface of the function above for performance of the PSO algorithm

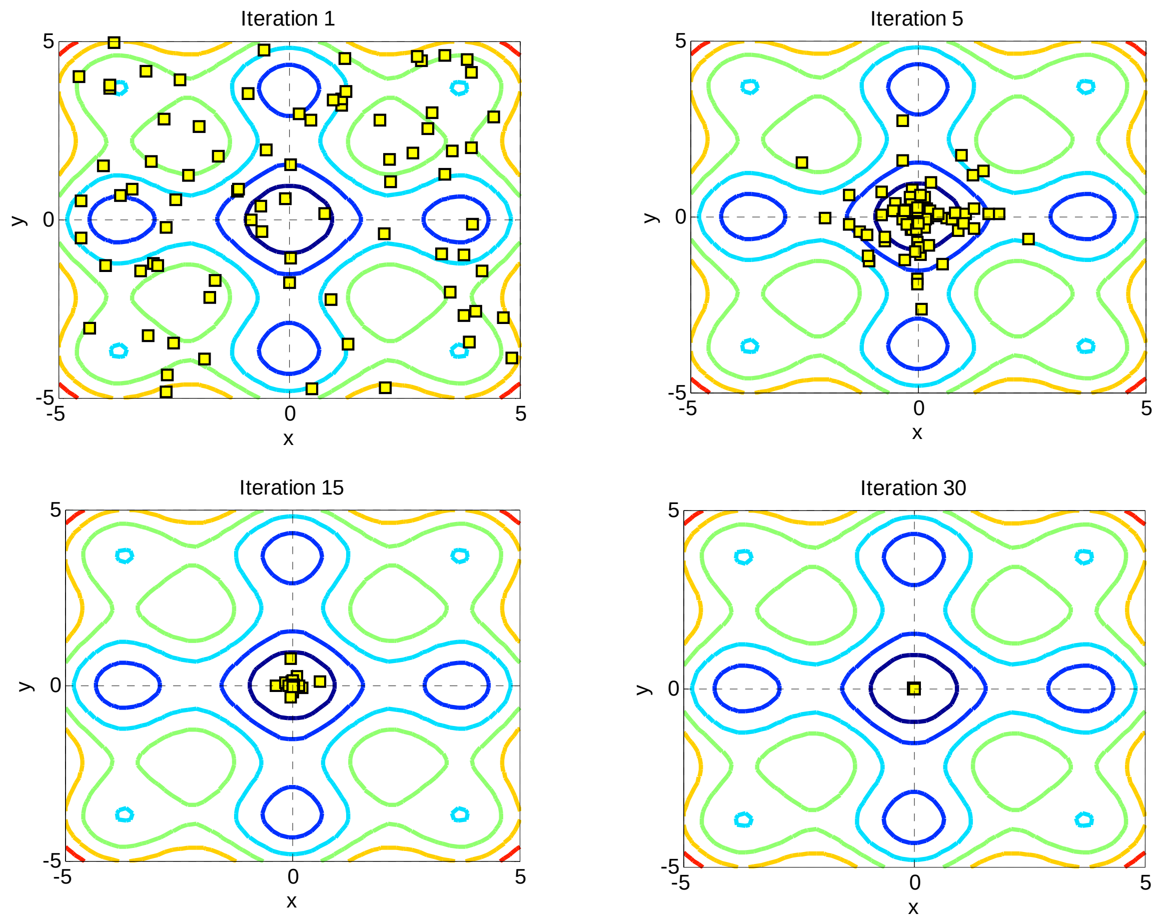

Figure 4.3 Selective Iterations of the Performance for the Example Function

4.3 Proposed Methodology

After the explanations given in the previous sections, now it is appropriate to explain the methodology used in this thesis to tune the AUKF filter using Particle Swarm Optimization for a specific application of FDI for a RW unit of a pico-sattelite ACS. This section will explain how to implement the optimization in the tuning problem mentioned in Chapter 1.

As explained in Chapter 1, the main focus of this thesis is to propose a methodology based on optimization algorithms to “systematically” tune filters and/or controllers for different purposes. In order to do that, as explained earlier, the main objective of using optimization algorithms is to minimize or maximize a value of a function by choosing the proper independent variables of the function. In FDI algorithms, the main idea and objective is to minimize the residuals in the estimations so that the estimations match the actual measurements (or the parameters of the system) perfectly. Combining these two ideas, the motivation came to use optimization algorithms to tune a Kalman Filter used for the purpose of FDI on a RW of an ACS of a pico-sattelite addressed in Ref. [17] and was developed in the SSDC lab. Also there was a need for an algorithm to be able to tune any filter or controller systematically and easily with a friendly user interface to adapt itself to the system with the minimum changes requires and tune the system giving the parameters/gains with the least human interference and time consumption.

With that in mind, the idea used in this thesis is to construct a modular optimization algorithm that can be used to tune filters/controllers for different purposes. To do so, the available code was divided into different subroutines as listed in Table 4.2.

Table 4.2 Different modules of the code and a brief description for each

Module	Description
Main	The centralized command centre for the whole code
Initialize	Initializes the model parameters used for tuning
Objective	Defines the objective function for the optimization
PSO	The core of the particle swarm optimization algorithm
Model	The model used in the simulations for optimization
Plotter	Plots all the required data and shows on the screen

More detailed description on each of the modules is as follows:

Main: the main module contains all the necessary information for a simulation to run and it also calls other modules within itself when necessary so that everything is done in time and properly to help get the desired results. Therefore, it is acting like a command centre for the whole code and simulation.

Initialize: in this work, Simulink model is meant when referring to a simulation; there typically is a need for pre-initialization of the system parameters so that the system will run and if there is any need for any change in a specific part of the system or parameter then this module helps managing that. This module also hold within itself time of occurrence and the severity of faults introduced to the system during the simulation. This means that this module can be later used for any other system to initialize the necessary parameters and faults properties, if required. This makes the code clean and easily adaptable to any other system.

Objective: this module is to maintain and evaluate the index of each particle in each iteration and the construction of the objective function is critically important because it tremendously influences the performance of the optimization algorithm. The inputs for this module are optimization variables and the simulation run time for the system model (i.e. Simulink Model) which is separately designed and attached to the optimization algorithm. The variables are fed to the system and the model is executed using those system parameters, then after the execution is done, the results are fed to the Objective module again and the objective function is formed and evaluate and the evaluation results are sent back to the PSO module so that the optimization process and continue.

One question that remains unanswered is “how to construct a desirable objective function based on the available data?” The answer to this question is simple and intuitive. As discussed before the main objective in the FDI algorithm is to minimize residual error and hence, using an optimization algorithm would help minimizing the residual error. Having said that, the objective function should be constructed based on the residual errors so that in each evaluation, the least value would be chosen and the algorithm continues to find the minimum among all. In this study, a combination of Room Mean Square Error (RMSE) and normalizers to give satisfactory results for the problem in question. There are few steps to take while constructing the objective function and if these steps are followed then the results could be guaranteed to be satisfactory with conditions applied.

1. First of all the general form of the objective function which is used for minimization of the error in the following form:

$$J = \sqrt{\sum_{j=1}^{k}\left(r_j \cdot \frac{1}{N} \sum_{i=1}^{N} (X_{j,i} - \hat{X}_{j,i})^2 \right)} \tag{4.9}$$

where r_j is the normalizer, $X_{j,i}$ is the actual value for the j^{th} variable, $\hat{X}_{j,i}$ is the estimated value for the j^{th} variable, k is the number of optimization variables and N is the number simulation samples. In this formula, the only thing that needs to be set is the value for normalizers. Normalizers are valuables that come into the formula to, as their name conveys, normalize the components and their influence on the whole function to keep the influence of each component same as others so that all the variables are optimized equally. In the first attempt, there was a proposal for using dynamic normalizers but the simulations proved that dynamic normalizers, at least the ones that were proposed, did not work properly. Because the idea was to keep the value in the unity range (between 0 and 1) and because the normalizers were changing in each iteration then the influence of each component would change automatically and hence, there was no control over what influence is being done by which component. Therefore, constant normalizers were then used and the value for each can be found using the next step.

2. To find a proper value for each normalizer there is a need for the system in question to be executed once so that it gives the designer an estimate on what is the order of each comonent. Then having this estimate the designer needs to keep all the components of the objective function in the same order. More description on this is given in the next Chapter where the results are presented and discussed.

PSO: this module is the core of the optimization algorithm. It is called from the main module and it reads some of the required information from the command sent to it from the main module while it was being called. These parameters which include the number of iterations, number of particles and also the number of duration of the simulation help PSO module to be dynamic in a sense that it is not rigid and it controllable from the outside environment (here the main module) and also within the PSO module there are some other parameters that need to be set. These parameters include the number of parameters that the algorithm needs to optimize. It tells the algorithms that how many dimensions the search-space will have. In addition, there is a range for each parameter and these ranges are set in this module. All ranges together create the search-space. Another set of parameters that are set in this module are the parameters for the PSO algorithm itself, which influence its performance and were earlier discussed in Eq. (4.5). In this thesis, these values are set for all simulation as listed in Table 4.3. The reason for this choice of values was discussed in the previous section. One other important factor here is how the boundaries are set and how the algorithm makes sure that particles do not move away from the search space. The answer is simple and intuitive; after all the positions and velocities are calculate in the algorithm, in the end there is a checking procedure of each particle's location to make sure that it is not beyond the set boundaries. If it is then the location is set to the nearest boundary and the velocity is changed to the opposite direction with the same magnitude. This way the algorithm guarantees that all particles remain in the search space while the simulations and calculations are in progress. The output of this module is the trend of the objective function through the optimization process and the best values found (optimal solution) for the problem in question.

Table 4.3 PSO parameters values for the simulations

Parameter	Value
w	1.0
c_1, c_2	1.0

Model: this part of the structure is a model designed by any other person. In here because the simulations are executed in the MATLAB programming language then the model is in the Simulink environment but there is a capability for any other executable file with inputs and outputs to be attached to this tuning/optimization modular structure. For that, this could be used for a variety of different systems designed in different environment and by different people. This configuration makes this approach a decentralized topology which means that not everybody needs to know what is going on in other parts of the system and each person in the team only needs to know what he is doing and what are the inputs and outputs of the module/model they are designing or working on. In this thesis, as described before, the model in use is a high fidelity RW [59], which is depicted in Figure 2.2.

Plotter: this module is that last module called in the main module to finally show the results on the screen and give a perspective of the performance of the algorithm as well as necessary figures and graphs required by the designer. This module, for sure, needs adaptation for different systems with different inputs and outputs because when the optimization variables are changed the figures needs to be modified to be consistent with the system and the requirements.

The flowchart of the all above-mentioned modules of the proposed structure is shown in Figure 4.4 below. In this figure, X is the vector for the optimization variables which are the variables or parameters needed to be tuned, R is the result of the simulation from the model vector which includes all the required results from the simulation and J is the objective function built and evaluated using the R vector from the simulation and sent to the PSO so that the optimization procedure is complete. These steps in the optimization block continue until the maximum number of iterations is reached and then the results are sent to the plotter module for plotting.

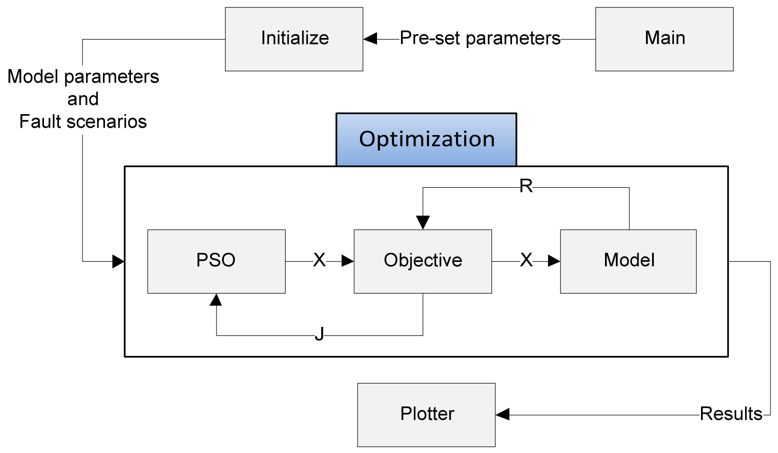

Figure 4.4 Flowchart of the Proposed Algorithm for Tuning

Now that the methodology is clearly explained, in the next chapter the simulations and results are presented to evaluate the performance of the proposed methodology.

5. Results and Discussions

Simulations of the FDI algorithm with the tuned AUKF were performed in MATLAB/Simulink to verify the performance of the proposed methodology. Numerical simulations are conducted on the high fidelity RW model presented in Chapter 2 with the bus voltage and BEMF constant being the Fault Parameters (FPs). In order to be able to compare the results with a manually tuned filter, the simulation parameters are chosen to be the same as the ones available in Ref. [17].

5.1 Simulation Setup

The high fidelity reaction wheel model shown in Figure 2.2 was used for simulations with the parameters listed in Table 5.1. Because of the high degree of non-linearity in the model, the AUKF algorithm was applied. The simulation was set up as shown in Figure 5.1.

Table 5.1 Ithaco - Type A - Reaction Wheel Parameters

Parameter	Value
Coulomb Friction (τ_c)	0.002 N.m
Viscous Friction (τ_v)	3.84×10^{-4} $N.m/rad/s$
Ripple Torque (B)	0.22
Cogging Torque (C)	0
Torque Noise Frequency (ω_a)	0.2 Hz
Jitter Angle (θ_a)	0.05 rad
BEMF (K_e) Nominal	0.029 $V/rad/s$
Bus Voltage (V_{bus}) Nominal	8 V
Driver Gain (G_d)	0.19 A/V
Num. of Motor Poles (N)	36
Input Filter Resistance (R_{IN})	2 Ω
Quiescent Bus Power (P_q)	3 W
Driver Bandwidth (ω_d)	9 rad/s
Voltage Feedback Gain (k_f)	0.5 V/V
Flywheel MOI (J_w)	0.0077 $\text{kg}\cdot\text{m}^2$
Over-speed Circuit Gain (k_s)	95
Max. Wheel Speed (ω_s)	680 rad/s

A control voltage trajectory is applied to the high fidelity Simulink model of the RW that serves as the plant. The outputs of the system are the wheel speed and current ω and i respectively. To simulate sensor measurements these outputs are then discretized using zero-order-holds with a sampling period of T_s. The control voltage trajectory is also discretized to simulate the discrete environment. All these discretized components are fed into the parameter estimation algorithm where a residual is generated as the difference between measured states and outputs of an analytical model of the RW that is running in parallel to the filter. Thus, the parameter estimates adjust such that the analytical model outputs match the measurements as best as possible. White noise is injected into the discretized outputs of the RW model to simulate measurement noise.

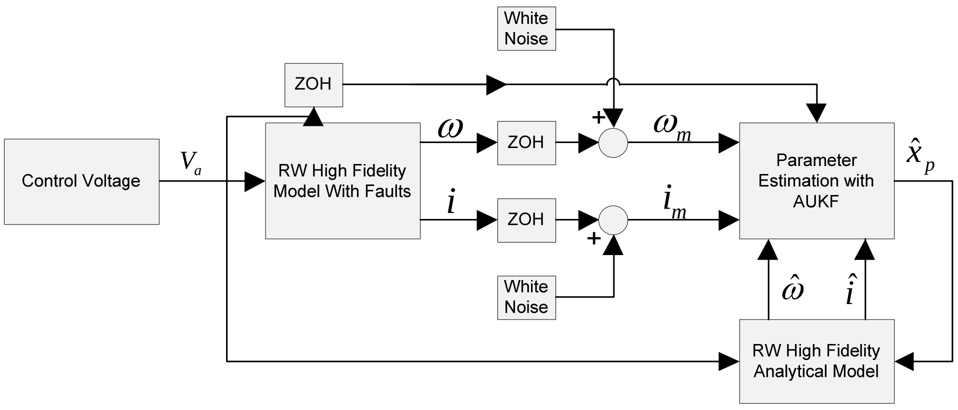

Figure 5.1 RW FDI Simulation Setup [17]

The white-noise signal power is calculated as follows,

$$P_v = 10\log_{10}\left(\frac{1}{T}\sum_{0}^{T}\left|w_{psd}\right|^2\right) \tag{5.1}$$

where w_{psd} is the power spectral density (PSD) of the white-noise. The nominal power spectral densities of the wheel speed and current measurements are 1×10^{-5} and 5×10^{-8} respectively resulting in -9.99 dB and -33.01 dB respectively of noise-power to be consistent with the simulation setup in Ref. [17] so that later on comparisons could be done.

Based on investigation done in Ref. [18], various experimental experiences with RWs on-board satellite missions have revealed the following potential sources of failure;

(i) Faults in the bus voltage

(ii) Faults in the motor torque/BEMF constant

As a result, the parameters being monitored here are the bus voltage and BEMF constant. In Ref. [17] the reason for this monitoring is mentioned to be that

"Changes in BEMF constant can be attributed to extreme temperatures in the windings that exceed the Curie temperature of the magnetic material in the motor resulting in a decrease in magnetism of the magnets. Furthermore, any blunt-force trauma imparted onto the magnets can degauss them. Bus voltage faults may occur as a result of things like cold-solder joints, loose wires, or failures in the power supply."

Regardless of the cause it is important to monitor these parameters to improve the performance of RWs. Simulations were initialized as listed in Table 5.2 below.

Table 5.2 Simulation Parameters for RW FDI

Parameter	Value
Sampling Period (T_s)	0.01 s
Simulation Time	4000 s
Window Size (N)	*
ζ	*
$\mathbf{R}_0$	$2\times10^{-4}\times I_{2\times2}$
$\mathbf{Q}_0$	$1\times10^{-5}\times I_{2\times2}$
$\mathbf{P}_0$	$1\times10^{-8}\times I_{2\times2}$
$\mathbf{Q}_{max}$	*
$\mathbf{R}_{max}$	*
$\hat{\mathbf{x}}_0$	$[8,0.029]^T$
κ	-1
n	2

The parameters in this table are exactly the same as the ones given in Ref. [17]. In this reference, the reason behind choosing each parameter is explained as follows. As can be seen in this reference the window size is selected to be large because, as claimed by the author, for parameter estimation it is assumed that parameters are not dynamic quantities for the most part, thus the problem that the filter might not be able to track the changes if the window is too large, is not an issue here. It is also mentined that this window size was found to be applicable to all the cases presented in Ref. [17] below for nominal measurement noise thoguh we only work with the most severe case in the cited reference. In this theis the selection of windows size is automatic and the value for this is an output from the optimization algorithm hence, there is no need for manual tuning and extensive simulation for a person to come up with a value and justify the performance. The UKF parameter κ is typically chosen as $\kappa = n-3$ for all simulations, where n is the number of estimates. In Ref. [17] ζ is selected to be 100 so that ε would remain below 1×10^{-2} and the author claims that the reason for that is because after doing extensive simulations, anything above that led to instability. In this thesis though, the value for ζ is determined automatically by the algorithm and the only thing that is set is the range for the value so that the algorithm can search for the best solution within the search-space. The voltage applied at the motor terminals is a sine wave with an amplitude of $5\ \text{V}$ and a frequency of $0.25\ \text{Hz}$. The reason that not all various fault scenarios considered in Ref. [17] are also considered here is because first of all, the most severe case is being studied here and if the algorithm works for this case then it, for sure, will work for other cases as well. Secondly, the space limit for each case would not let the presenation of the work to be complete hence, it was decided to only present the results for the most severe case and then dicuss the results and performance for that case only. A high severity fault is considered as changes that are $\geq 20\ \%$, while low severity faults constitute changes $\leq 15\ \%$. The severe case cosists of a severe BEMF constant faults plus a sever bus voltage profile with each parameter fault occuring out of phase with the other. The profile is shown in Table 5.3.

Table 5.3 Severe BEMF Constant and Bus Voltage out of Phase profile

$time_{V_{bus}}$ (s)	V_{bus}	$time_{k_m}$ (s)	k_m
$t<1000$	8	$t<500$	0.029
$1000\leq t<2000$	6	$500\leq t<1500$	0.02
$2000\leq t<3000$	5	$1500\leq t<2500$	0.013
$t\geq 3000$	7	$t\geq 2500$	0.029

The performance of the system identification is performed using the root-mean-square (RMSE) of the estimation error calculated as follows,

$$e_k = \mathbf{x}_p - \hat{\mathbf{x}}_p \tag{5.2}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}[e_k]^2} \tag{5.3}$$

these performance indices are used for all simulation results to quantify the accuracy of the parameter estimates while the FDI performance is analyzed based on detection, isolation, and identification times. Figure 5.2 shows the input voltage profile applied to the RW and Figure 5.3 depicts RW outputs up to the $200\ \text{s}$ mark.

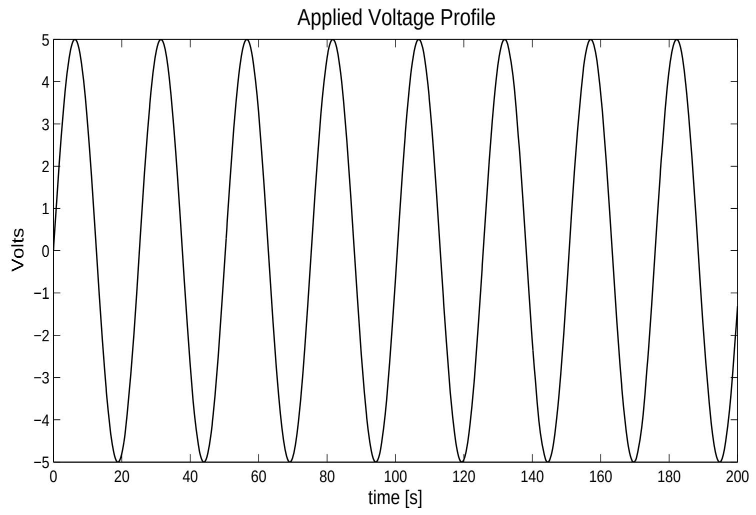

Figure 5.2 Reaction Wheel Applied Voltage Profile

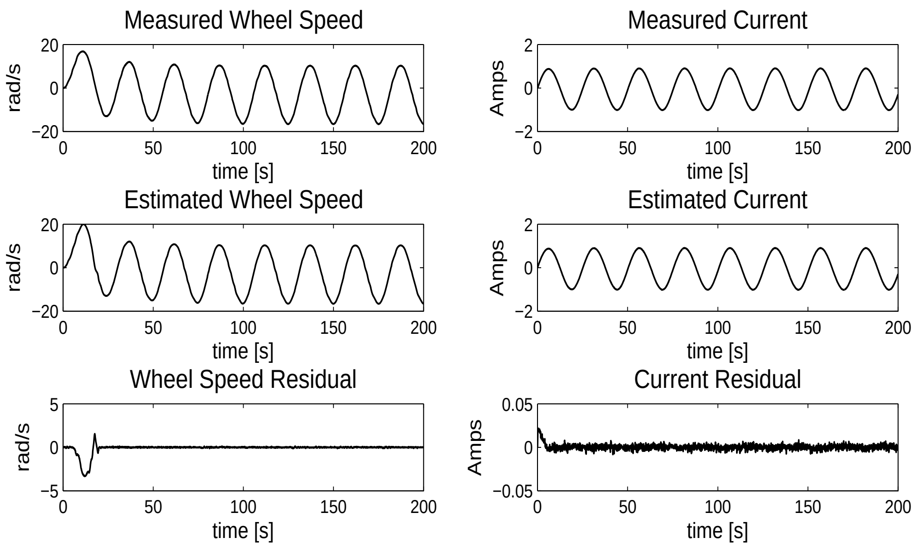

Figure 5.3 Reaction Wheel States for First 200 s

As discussed before, the main idea behind implementing an automated algorithm for tuning the filter lies in the minimization of residual errors. Here, 4 parameters with residuals available, namely Wheel Speed, Current, Bus Voltage and BEMF. Now that these parameters are known there is a need for an objective function to be formed and constructed based on the available parameters. To form the objective function, Eq. (4.9) is used and the following equation is constructed based on that,

$$C_1 = r_1 \cdot \frac{1}{N}\sum_{i=1}^{n} e^2_{wheel\_speed} \tag{5.4}$$

$$C_2 = r_2 \cdot \frac{1}{N}\sum_{i=1}^{n} e^2_{current} \tag{5.5}$$

$$C_3 = r_3 \cdot \frac{1}{N}\sum_{i=1}^{n} e^2_{bus\_voltage} \tag{5.6}$$

$$C_4 = r_4 \cdot \frac{1}{N}\sum_{i=1}^{n} e^2_{BEMF} \tag{5.7}$$

$$J = \sqrt{C_1 + C_2 + C_3 + C_4} \tag{5.8}$$

where e is the residual for each component from Eq. (5.2), C_i is a component for each parameter and r_i is the normalizer coefficient that we talked about earlier in Chapter 4. For all the cases studied in this thesis, these normalizers have the values shown in Table 5.4. These values are obtained using the procedure explained in the Chapter 4. The simulation was executed once and order of each parameter and residual was estimated. Then these values were set so that all parameters influence the objective function equally and the results are satisfactory for all parameters with the least residual error.

Table 5.4 Normalizer Values for Objective Function of the PSO algorithm

Normalizer	Value
r_1	2
r_2	2000
r_3	10
r_4	10^6

There are two noise levels involved in the first case. As referred to by Ref. [17] the medium and high noise levels are applied to the system and the performance of the algorithm in tuning the filter is evaluated. This case consists of attempting to estimate faults in both parameters (BEMF and Bus Voltage) where changes in one parameter occur out of phase with changes in the other one. It is expected that the BEMF constant will be affected by large fluctuations in bus voltage even when the estimate has settled, and that the bus voltage estimates will be fairly robust against changes in BEMF constant as mentioned in Ref. [17]. The fault period begins at $500\ \text{s}$ and ends at $3000\ \text{s}$, during this time the faults are injected into the RW system out of phase. Outside of this time range the parameters return to their nominal values. The parameters for medium noise level are listed in Table 5.5 and for the high noise level the parameters are listed in Table 5.6. Those parameters with the (*) sign in these two tables are the ones being tuned for the AUKF filter proposed in Ref. [17] and in the section for the future work there is a suggestion for automating the procedure of tuning the filter hoping for better performance and less time consumption by the designer. Performance of the filter is presented for different scenarios of optimization for performance evaluation.

Table 5.5 Simulation Parameters for Medium Noise RW FDI

Parameter	Value
Wheel Speed Noise Power	$0.0065\ dB$
Current Noise Power	$-23.01\ dB$

After evaluating the performance of the system with these two noise levels there is "Extreme Noise" introduced to the system and then the performance of the filter is compared with the two sets of gains, one for the gains presneted in Ref. [17] for robustness evaluation of the filter and one from the PSO-based methdology proposed in this thesis. The reason that this is called "Extreme" is because it is mentioned in Ref. [17] that these noise leveles are the nominal noise levels that the system could sustain.

Table 5.6 Simulation Noise for High Noise RW FDI

Parameter	Value
Wheel Speed Noise Power	$10\ \text{dB}$
Current Noise Power	$-13.01\ \text{dB}$

Simulations were executed on a PC in the SSDC lab with the specification listed in Table 5.7. The specifications of the PC that the simulations were executed on are specifically important because it affects the execution time directly. If the computer is fast, the execution time will be less and vice versa.

Table 5.7 Specification of the PC for Simulations Executions

Parameter	Value
Processor	Intel® Core™2 Quad CPU 2.66 GHz
Installed Memory (RAM)	4.00 GB
System Type	64-bit Operating System
OS	Windows 7 Professional with SP1

5.2 Effect of Number of Particles

In order to evaluate the performance of the proposed methodology and investigate the influence of the optimization parameters such as number of iteration, number of particles and the region or the search-space on the performance, few cases are studied with different scenarios. For these investigations, the medium and high noise levels are considered as in Ref. [17] to study the performance of the filter and to give a perspective on the difference between a manually tuned and a PSO-tuned filter. The region effect is not studied in this thesis because 1) the region of the search-space chosen for these simulations is big enough to guarantee that there is no compromise in the results obtained from the algorithm. Therefore, if the algorithm was to get trapped in local minima then because of the wide and big enough search-space there was a colossal chance for particles to be trapped. 2) here there are sets of values and not only one value or variable being optimized, simulations prove that there might be more than one solution to a specific problem with a reasonable deviation from the norm. Meaning that there could be different sets of variables or parameters being optimized in the algorithm that would result in almost the same value for the objective function. This is because these parameters influence each other as well as the overall performance of the filter. Hence, changing one of them could cause change on the others while maintaining the same objective function values.

5.2.1 Case 1 – 40 Iterations and 10 Particles

In this section, simulation results for 10 particles in 40 iterations are presented. The search-space for the optimization is as listed in Table 5.8. A question may arise on "how to choose the search space?". The answer would be that, for this type of solution there usually is a solution available in the literature and the objective is to improve the existing solution. Hence, starting from the available data and expanding the range for each parameter around the existing solution would be the approach on how to choose the search-space. Results for medium noise level are shown in Figure 5.4 to Figure 5.9 and results for the high noise level are presented in Figure 5.10 to Figure 5.15. The objective function trend and each individual component of the objective function trends are shown in Figure 5.16. In addition, the convergence trend for the optimization variables is depicted in Figure 5.17. Results for the parameters as well as more complementary information for the simulation are given in Table 5.9. There is a new acronym introduced in Table 5.9 and the other tables corresponding to the optimization results called "Objective Function Component Performance Index" or "OFCPI" this terminology is defined to gives further numerical values in addition to the visual representations of the objective function components values so that the comparison between different cases with different optimization parameters could be made more practically based on numbers and performance indices. The values for these indices are the same as the ones calculated in Eq. (5.4) to Eq. (5.8). More discussion on each result is given at the end of presentation of each individual case and then at the end of each section an overall discussion on the performance differences and the corresponding causes are provided.

Table 5.8 Search-Space for Simulations

Parameter	Range
$R_{\max}$	$[1, 10^4]$
$Q_{\max}$	$[1, 10]$
ξ (AFF)	$[0, 10^6]$
N (*Window size*)	$[1, 200]$

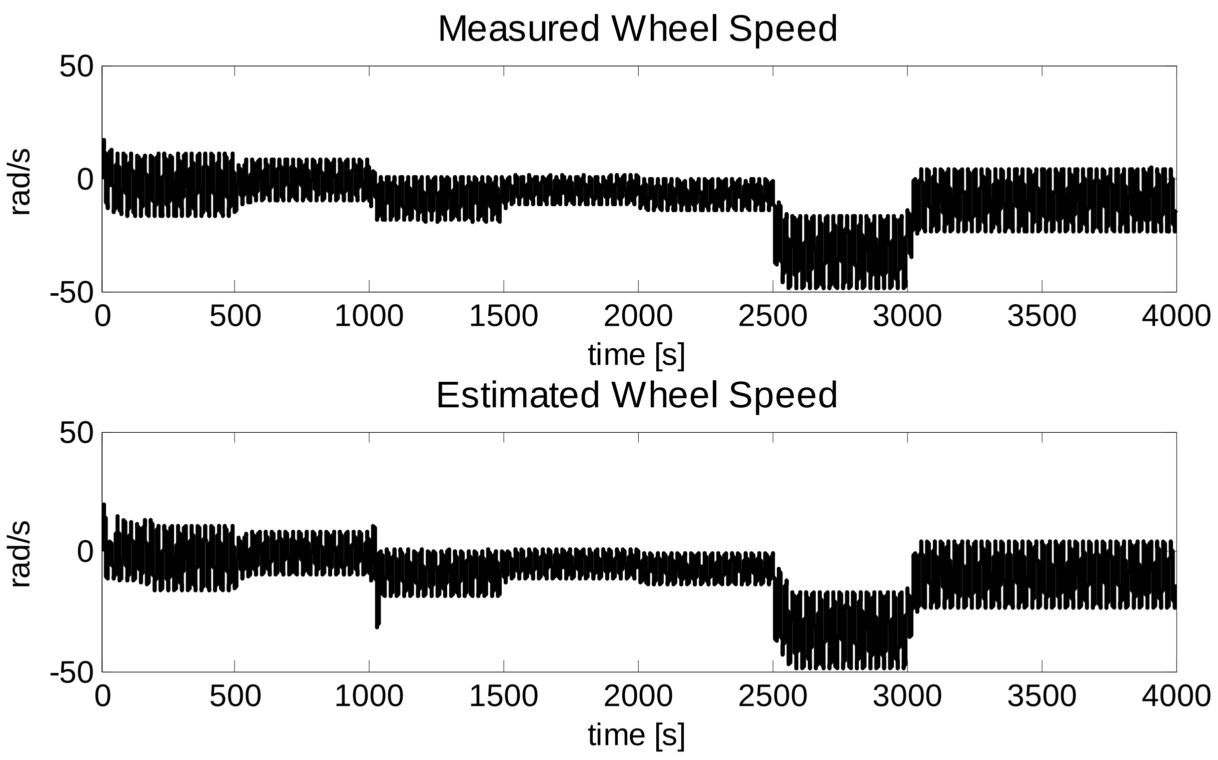

Figure 5.4 Case 1 - Medium Noise - Wheel Speed Measurements vs. PSO Estimates

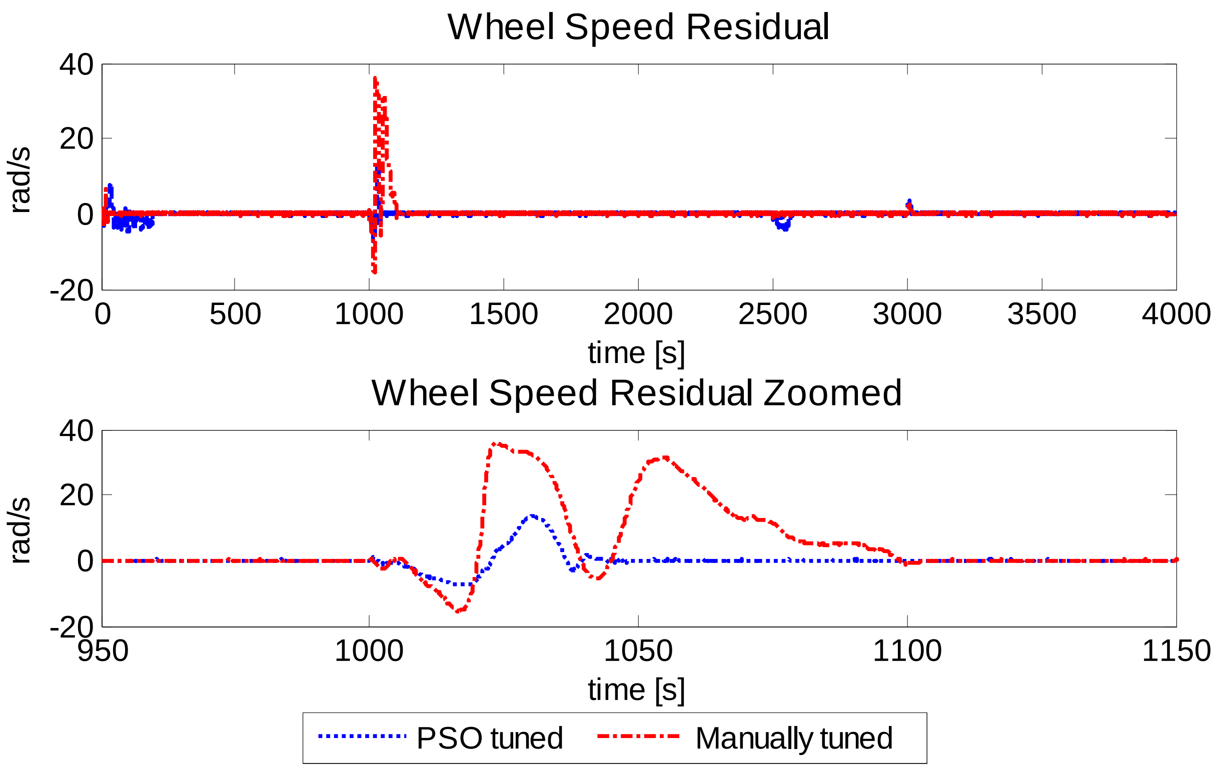

Figure 5.5 Case 1 - Medium Noise - Wheel Speed Residual Comparison

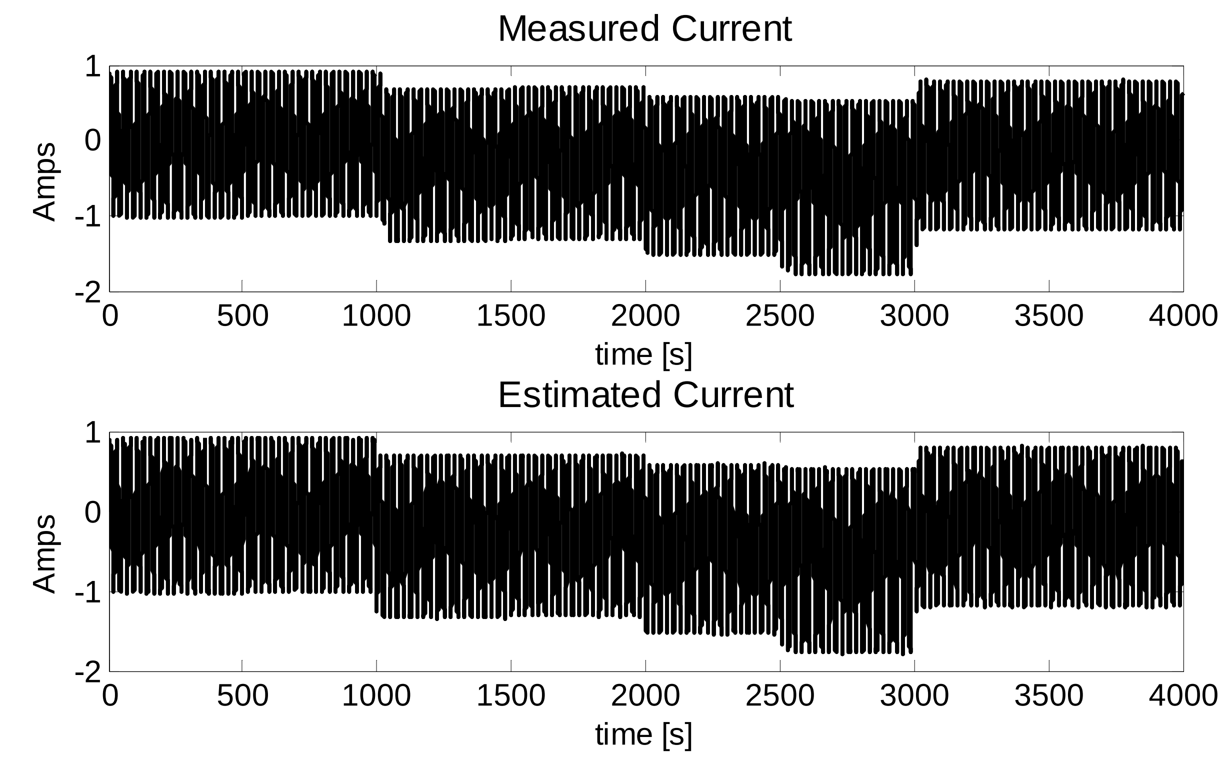

Figure 5.6 Case 1 - Medium Noise - Current Measurements vs. PSO Estimates

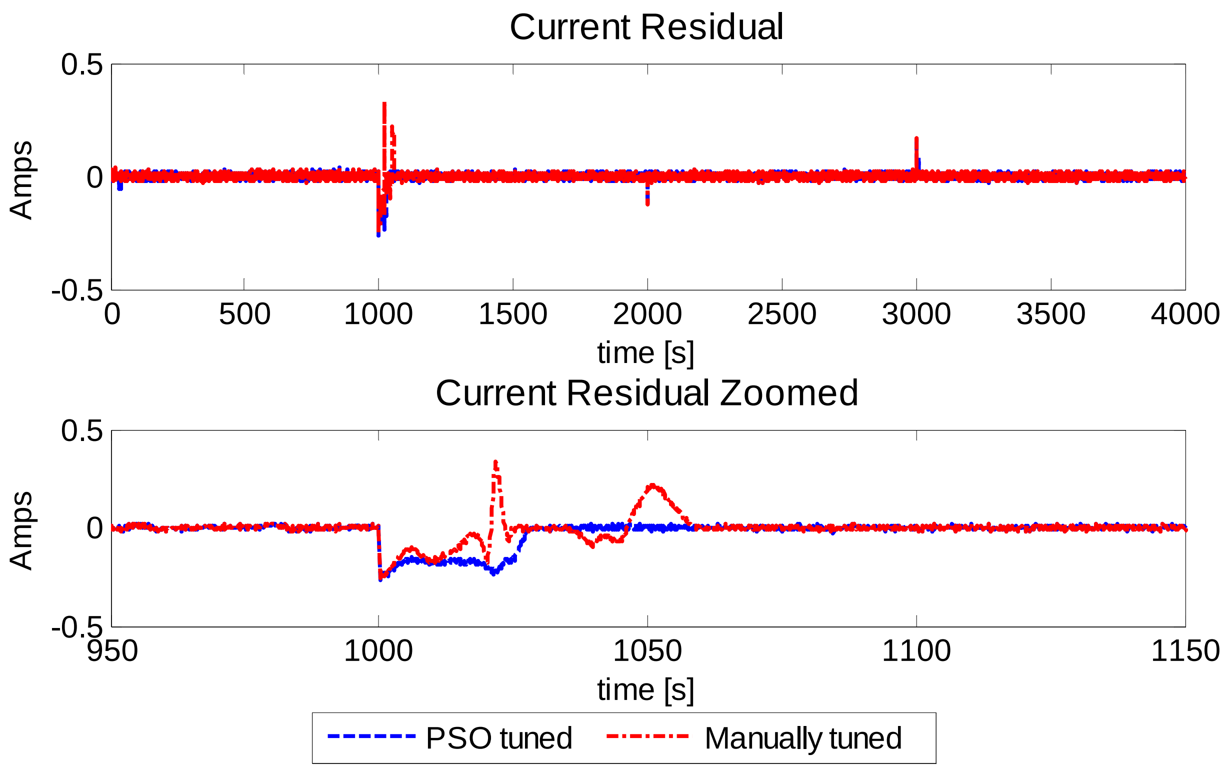

Figure 5.7 Case 1 - Medium Noise - Current Residual Comparison

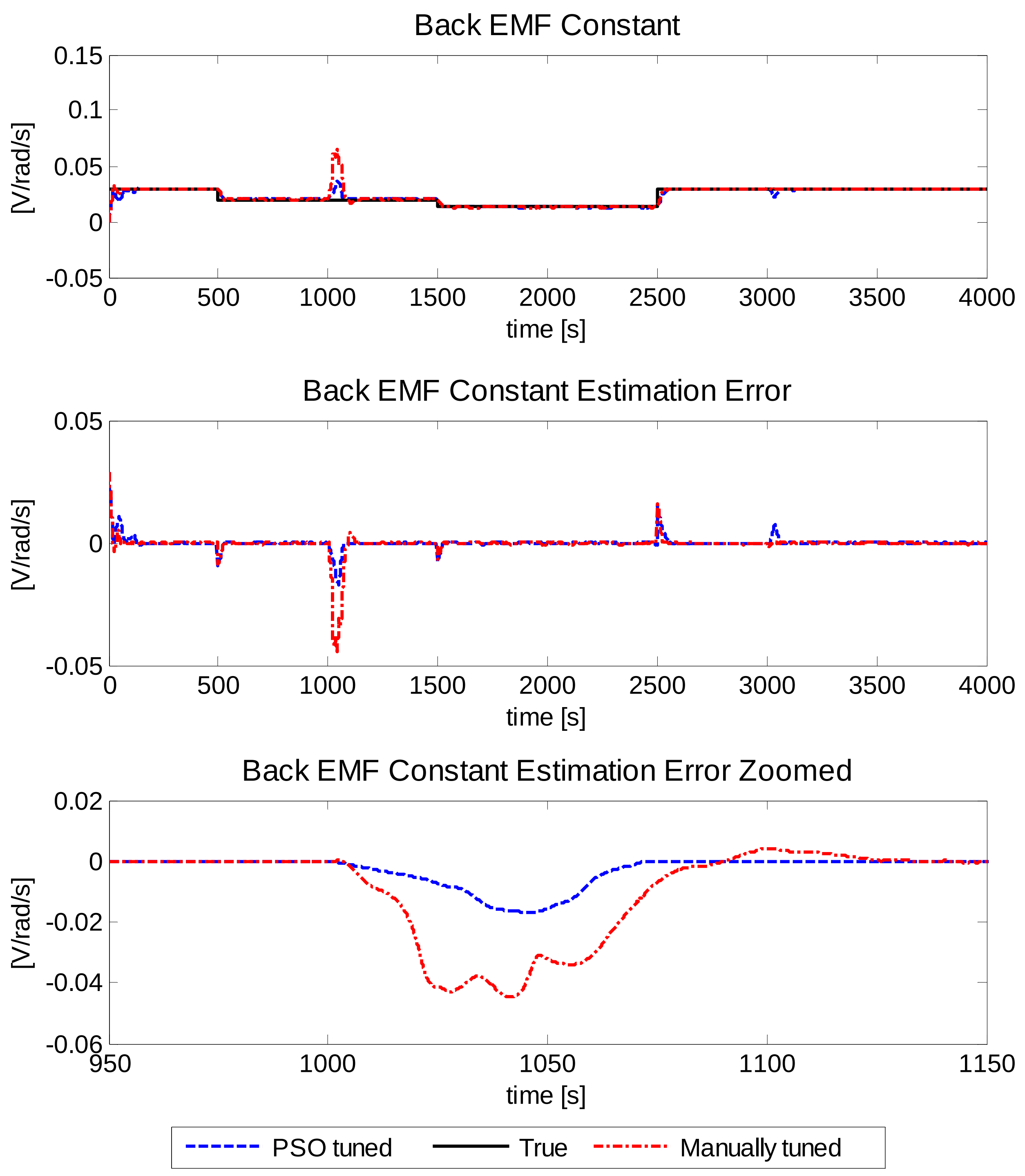

Figure 5.8 Case 1 - Medium Noise - BEMF Constant Estimates

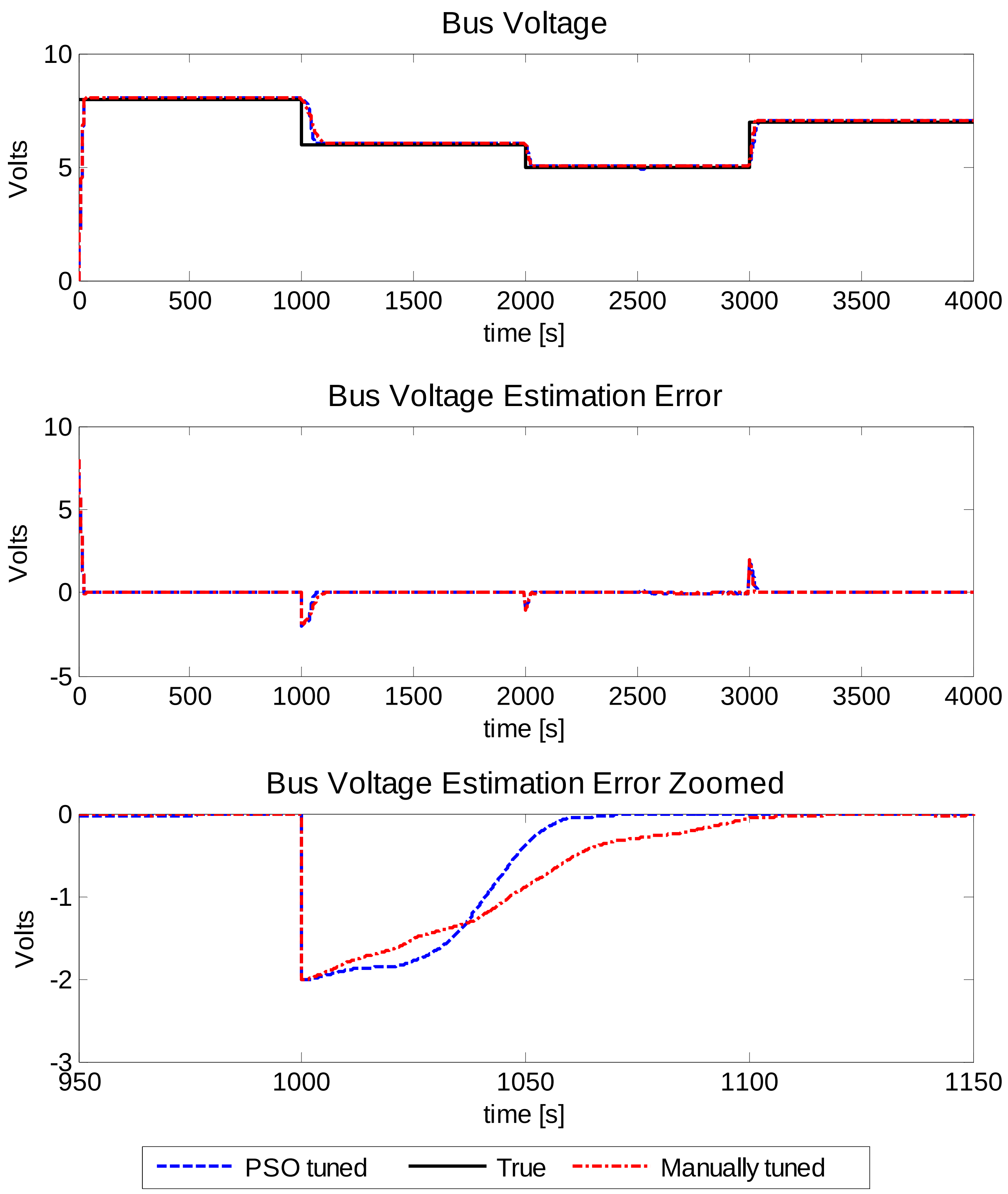

Figure 5.9 Case 1 - Medium Noise - Bus Voltage Estimates

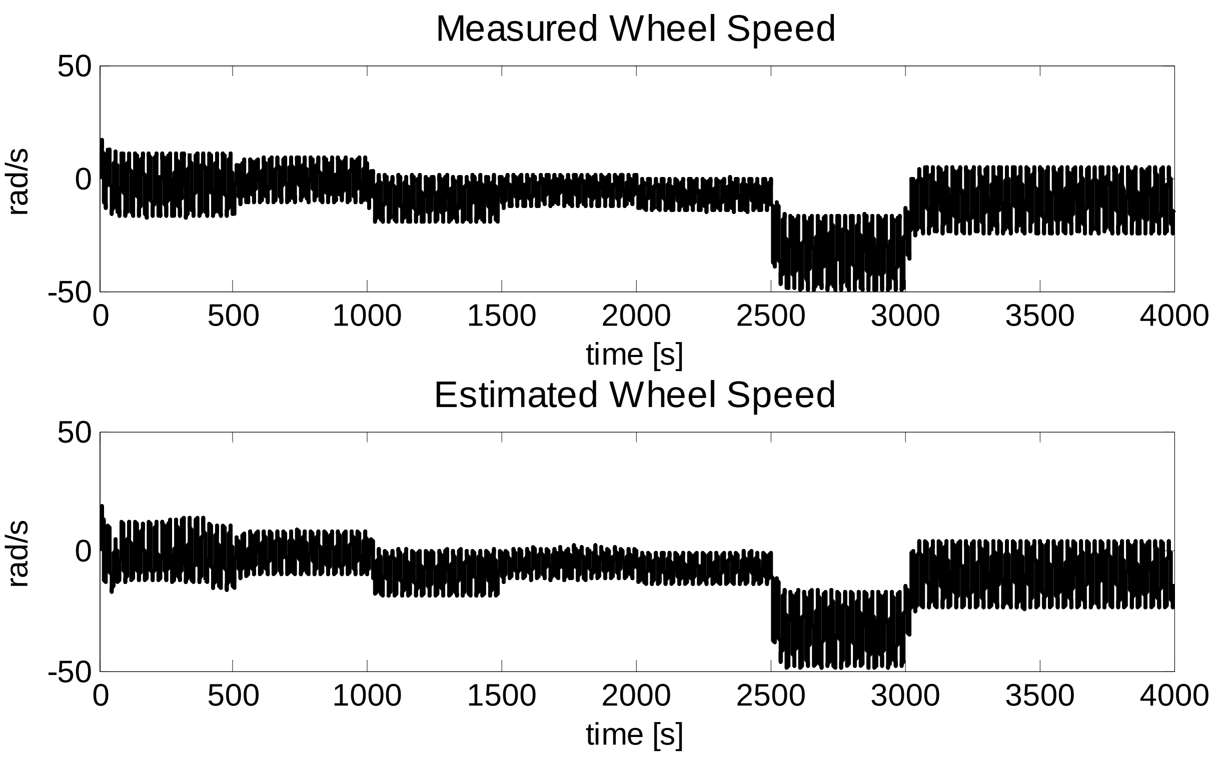

Figure 5.10 Case 1 - High Noise - Wheel Speed Measurements vs. PSO Estimates

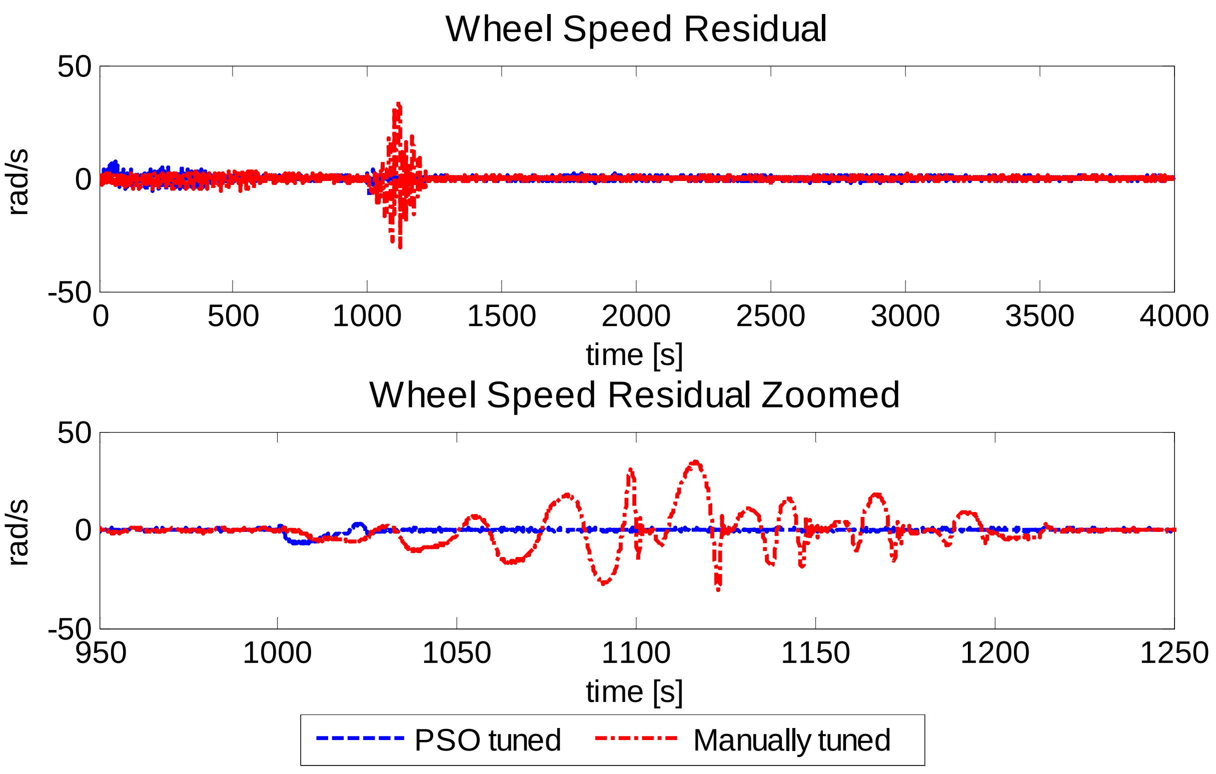

Figure 5.11 Case 1 - High Noise - Wheel Speed Residual Comparison

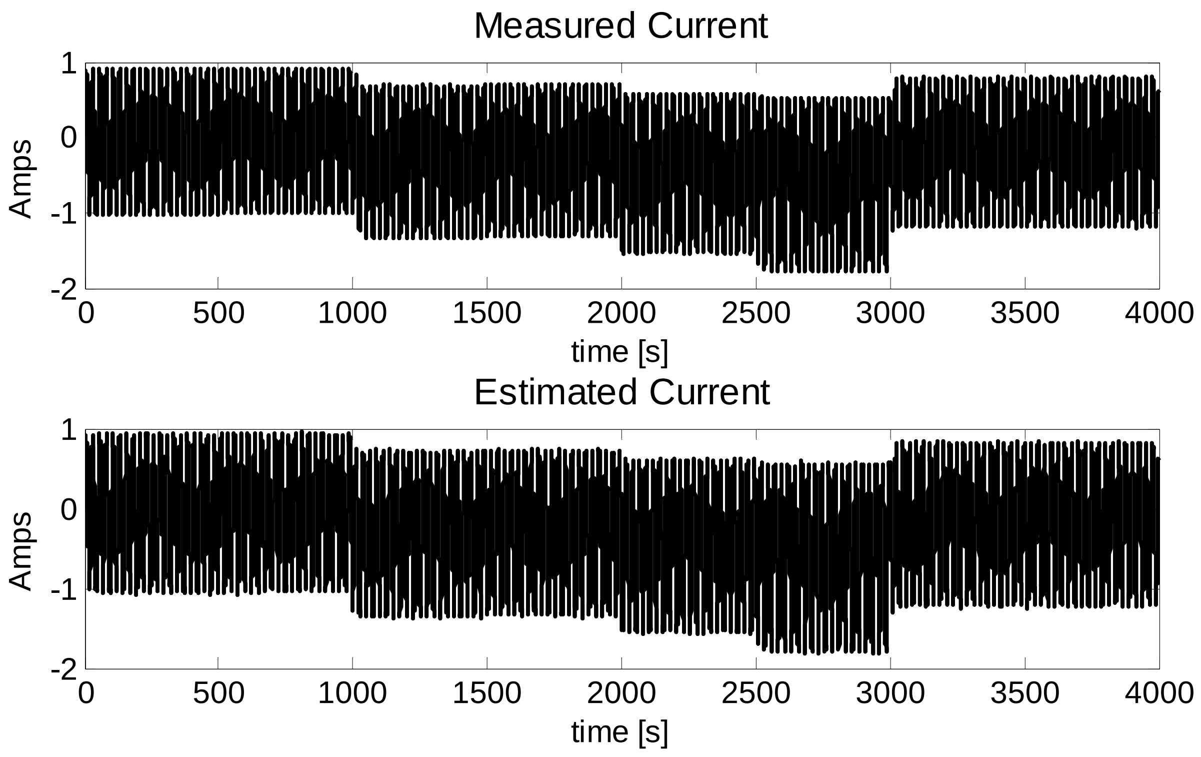

Figure 5.12 Case 1 - High Noise - Current Measurements vs. PSO Estimates

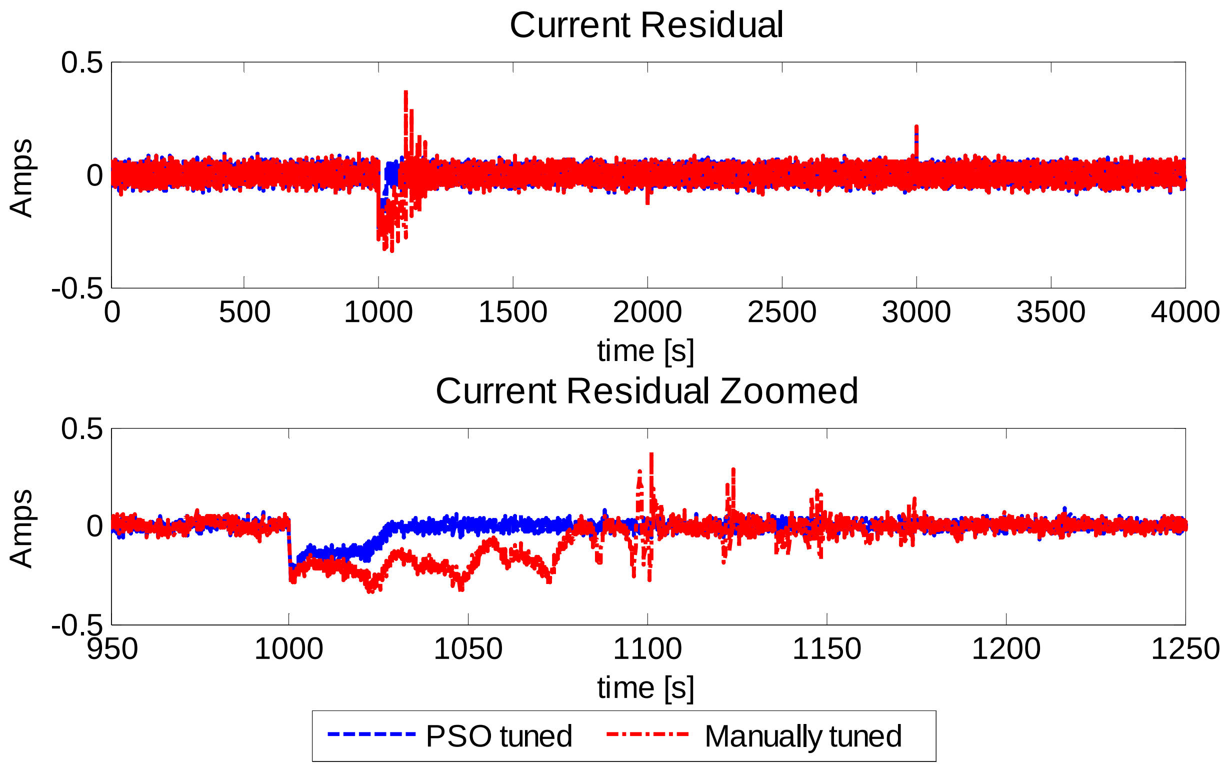

Figure 5.13 Case 1 - High Noise - Current Residual Comparison

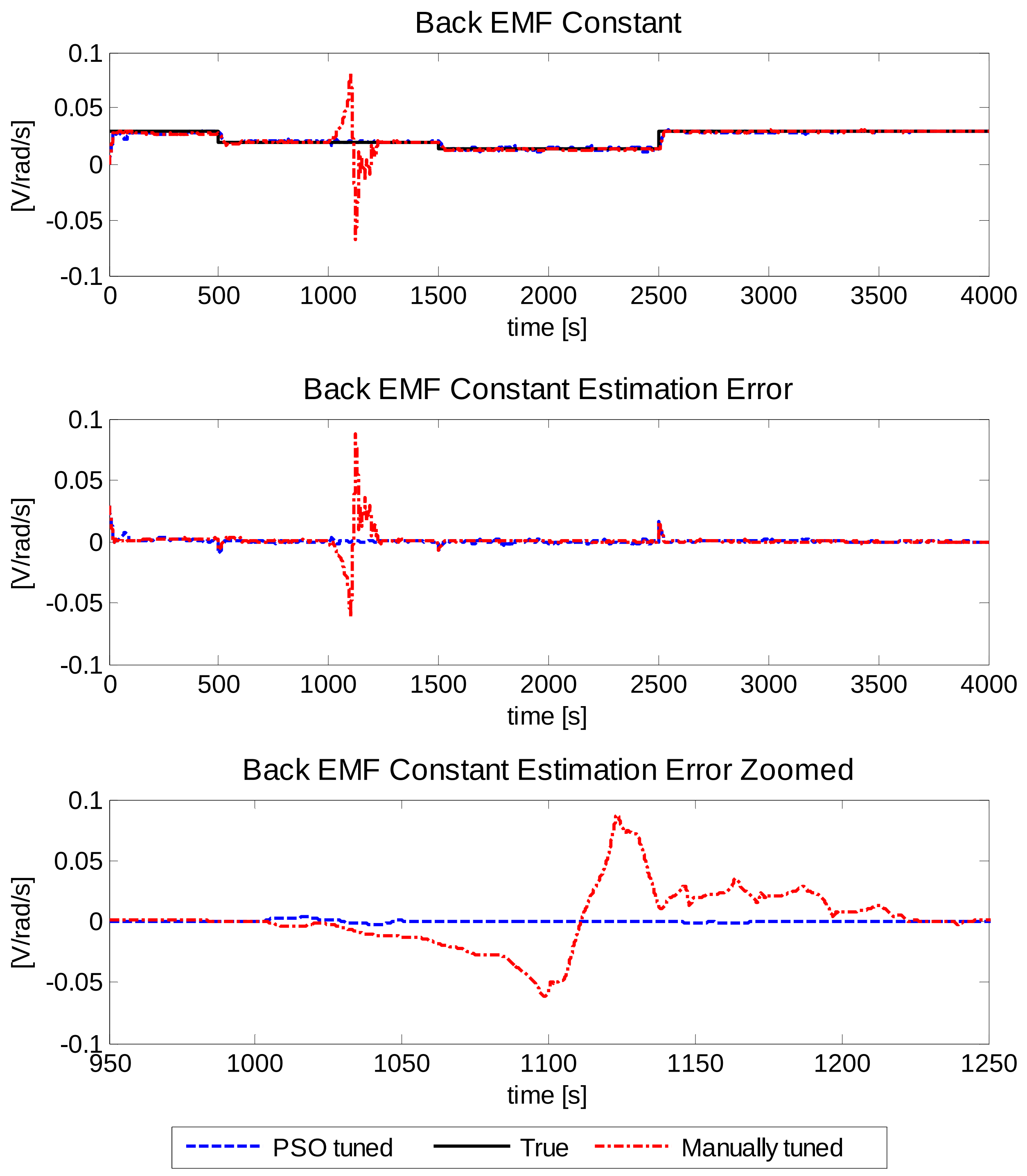

Figure 5.14 Case 1 - High Noise - BEMF Constant Estimates

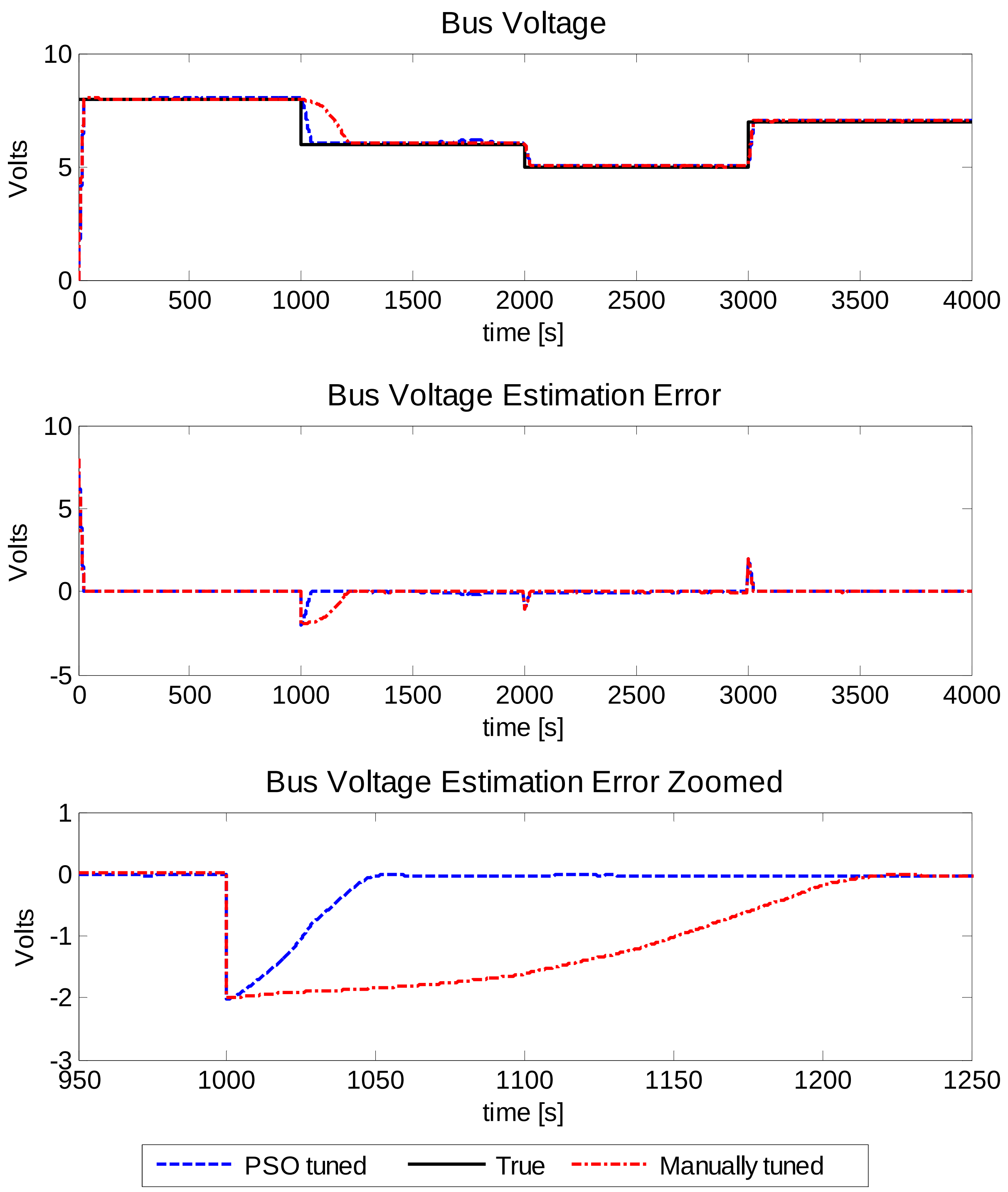

Figure 5.15 Case 1 - High Noise - Bus Voltage Estimates

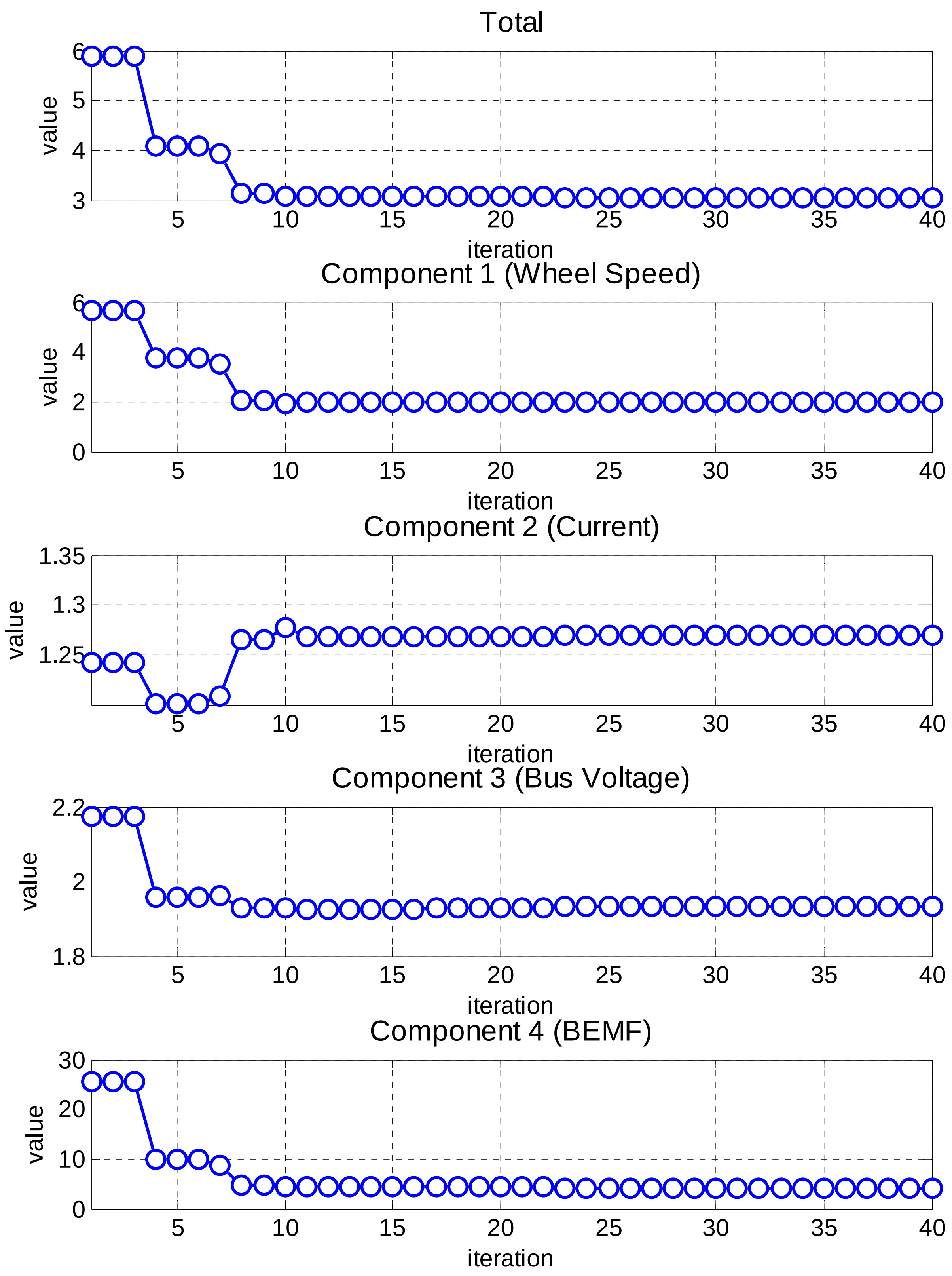

Figure 5.16 Case 1 - Objective Function Trend with Different Components Breakdown

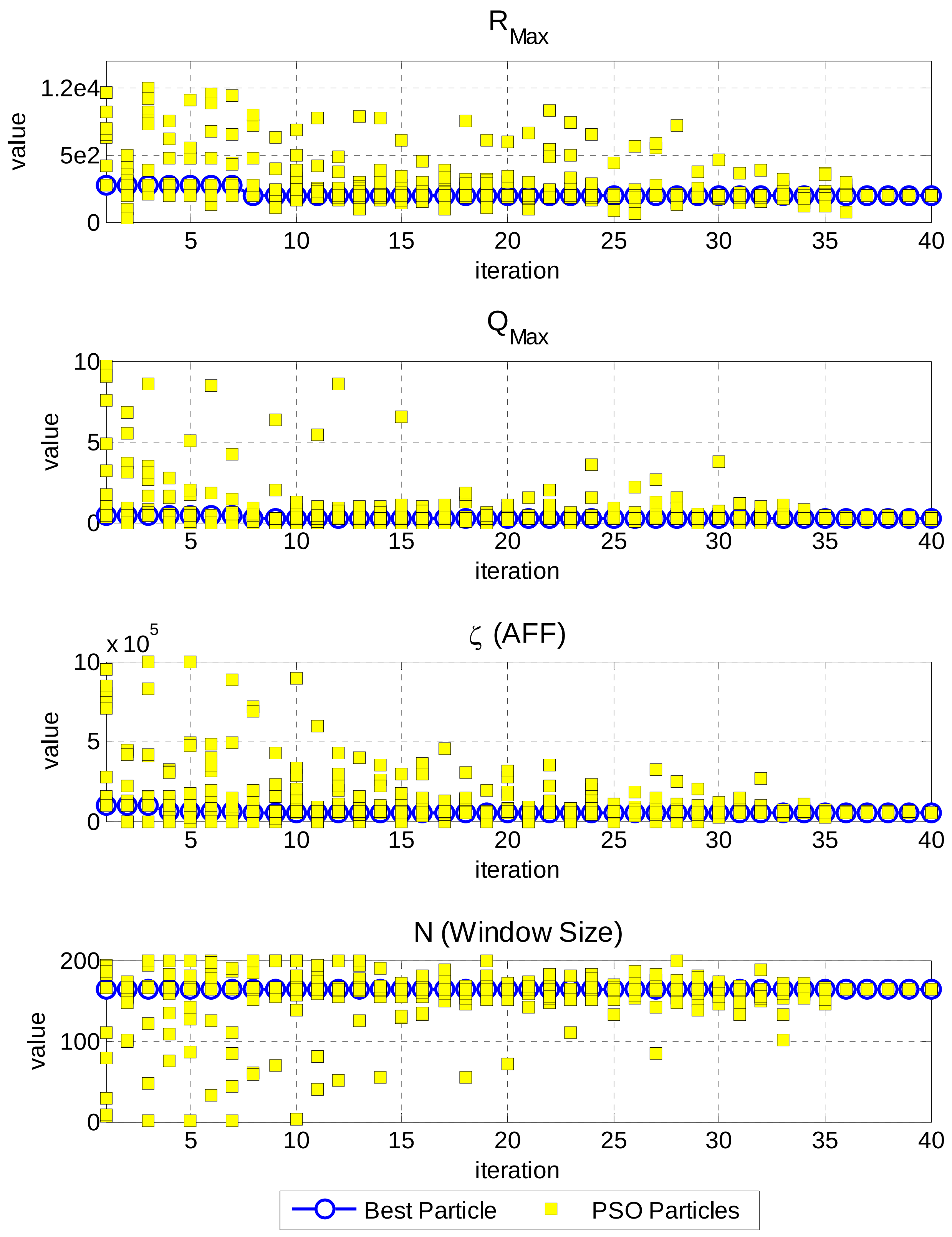

Figure 5.17 Case 1 - Parameters Convergence to the Global Optimum Trend over Iterations

Table 5.9 Case 1- Optimization Results

	Medium	**High**
Parameter	**Value**	
Execution time	19.34 hrs	
$R_{\max}$	1897.1	
$Q_{\max}$	0.2565	
ξ (AFF)	52765	
N $(Window\ size)$	164	
Total OFCPI	3.0686	3.0572
Wheel Speed OFCPI	1.2309	1.9821
Current OFCPI	0.8289	1.2689
BEMF OFCPI	1.9239	1.9308
Bus Voltage OFCPI	5.4327	4.1648

By investigating the results in Figure 5.4 to Figure 5.15 it is evident that the overall performance is tremendously improved and a proof for that is the decrease in residual for all the parameters estimated by the filter. The spikes are gone and/or decreased in elevation and the tracking of the parameter by the filter estimates is faster and there is less delay in tracking for all the parameters. One important factor to keep in mind here is that although the objective function is structured in a way so that the residuals for the high noise level are minimized, the results for the medium noise level prove that the gains found by the algorithm improve both medium and high noise level estimations. In Figure 5.17 for each iteration, all the particles are shown on the graph with yellow squares representing each particle and also white circles representing the best of the group in each iteration. The bests in each iteration are then connected to show the trend in the convergence of the particles toward the best of the group in each iteration. Overall, as the iterations forge ahead, all particles move toward the best position found up to the iteration. In some parameters, all particles converge to one point whereas in some parameters there still are some particles that are moving around to find a better position. The reason is that sometimes some particles, due to their inertia or momentum, jump over the solution. This can be solved by either using an inertia weight or increasing the number of iterations but overall it will not affect the performance of the algorithm because the best of the group will eventually be outputted as the best solution found. It is evident that after 35 iterations, all particles converge to a solution in the search-space that is the best solution found by the algorithm. Figure 5.16 shows objective function trend as well as each of its components trends so that the designer can get a sense of how the algorithm is behaving and how the objective function is being minimized. It also indicates what the influence of each component on the total objective function is. Another application for this figure is for determining the normalizers' values as discussed before. This figure gives you the order of each component and by setting proper values for the normalizers, it is possible to keep all the components in the same order and guaranteeing that each component is influencing the total objective function equally.

5.2.2 Case 2 – 50 Iterations and 5 Particles

In order to investigate the influence of number of particles on the performance of the algorithm, another set of executions were simulated of which only one case is presented here with figures and the rest are only presented in the table form. The search-space is the same as Table 5.8. Results for the medium noise level are shown in Figure 5.18 to Figure 5.22 and results for the high noise level are presented in Figure 5.23 to Figure 5.28 . The objective function trend and each individual component of the objective function trends are shown in Figure 5.29. In addition, the convergence trend for the optimization variables is depicted in Figure 5.30. Results for the parameters as well as more complementary information for the simulation are given in Table 5.10.

As can be seen from Figure 5.18 to Figure 5.28 the results are almost the same in comparison with the results from case 1 and but still superior to the results from the manually tuned filter. This execution was done for 50 iterations and the reason was to give the algorithm enough time so that all particles converge to the found solution. By investigating the particles convergence in Figure 5.30, it is evident that although the number of iterations is set to be higher than the number of iteration with 10 particles but all particles have converged to the solution after 38 iterations. It proves that when the number of particles is decreased, particles need more time to search the space and settle to the solution. This is mainly because the group performance of the swarm is less powerful with the smaller number of members. However, in the end the total OFCPI for both cases is almost similar, meaning that both cases after convergence provide satisfactory, near-optimal solutions.

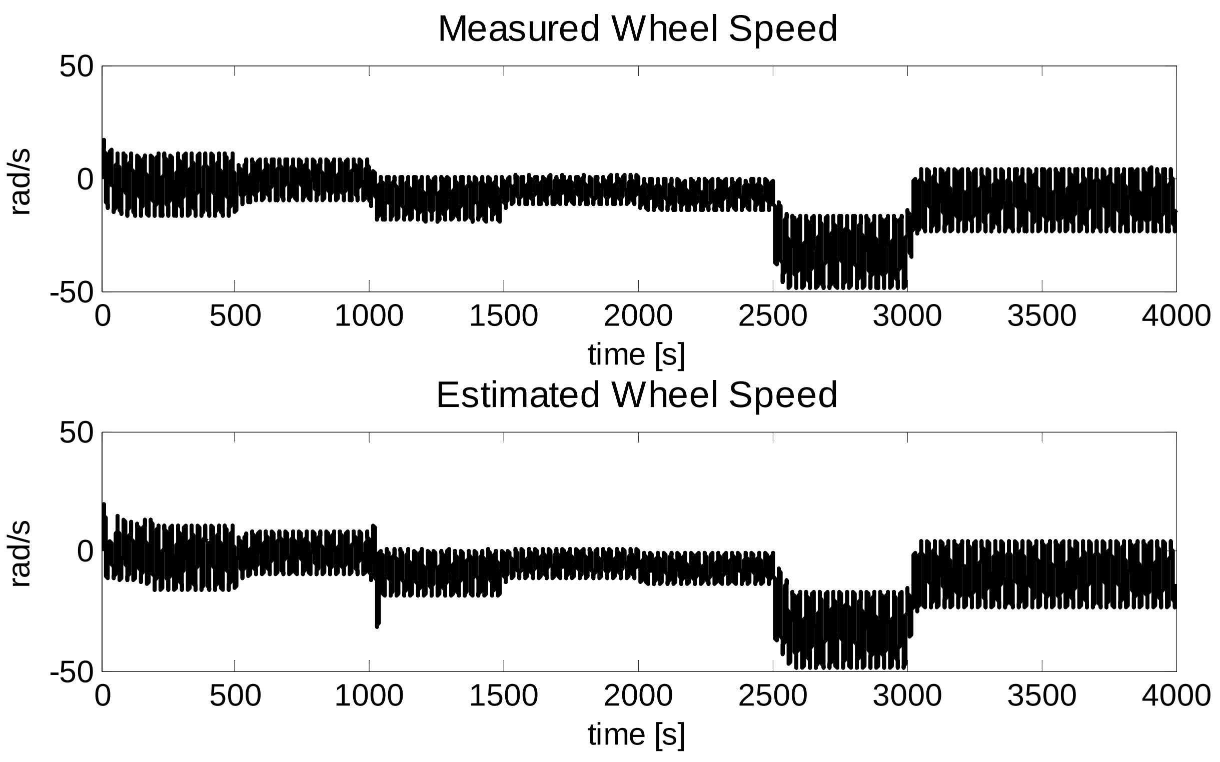

Figure 5.18 Case 2 - Medium Noise - Wheel Speed Measurements vs. PSO Estimates

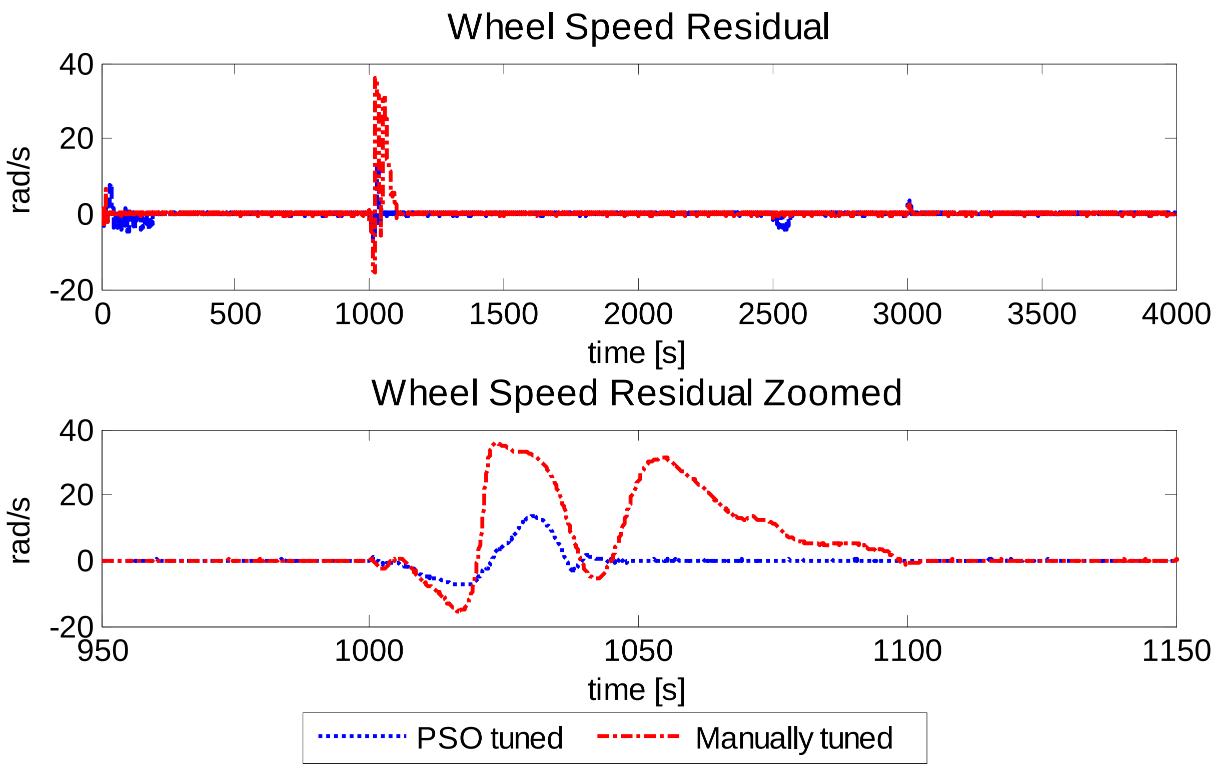

Figure 5.19 Case 2 - Medium Noise - Wheel Speed Residual Comparison

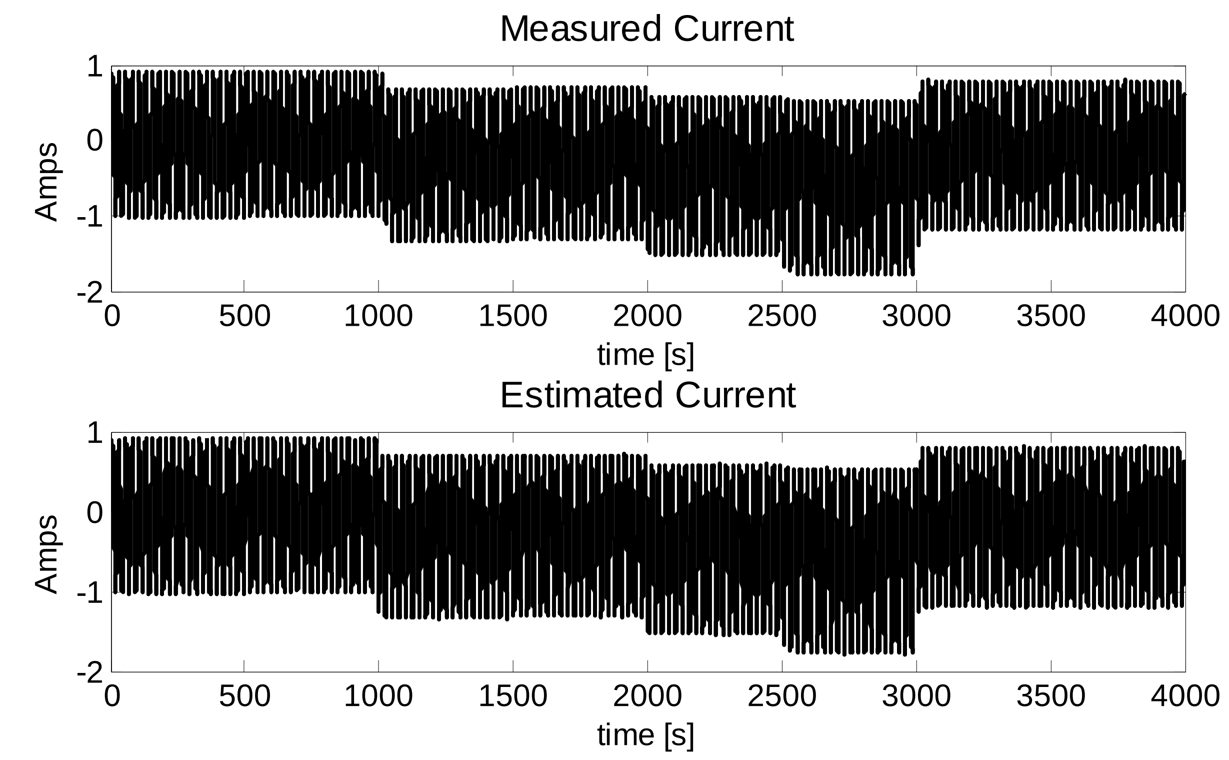

Figure 5.20 Case 2 - Medium Noise - Current Measurements vs. PSO Estimates

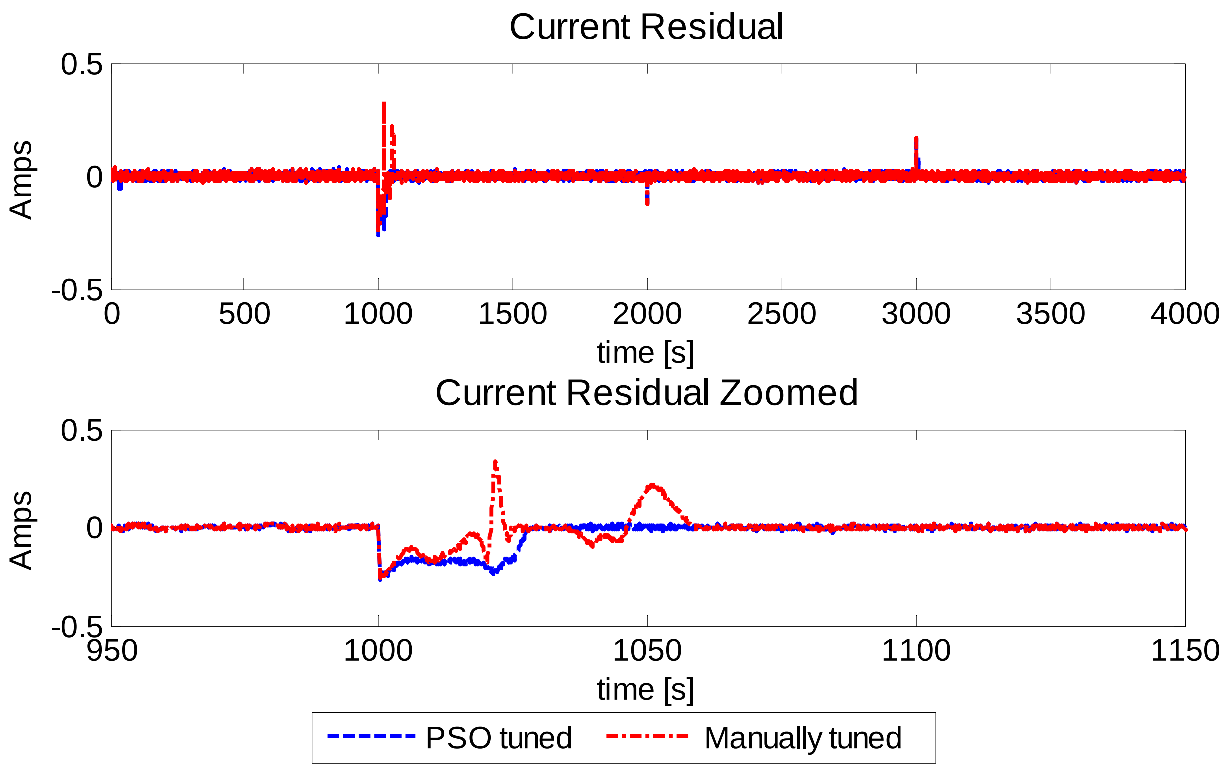

Figure 5.21 Case 2 - Medium Noise - Current Residual Comparison

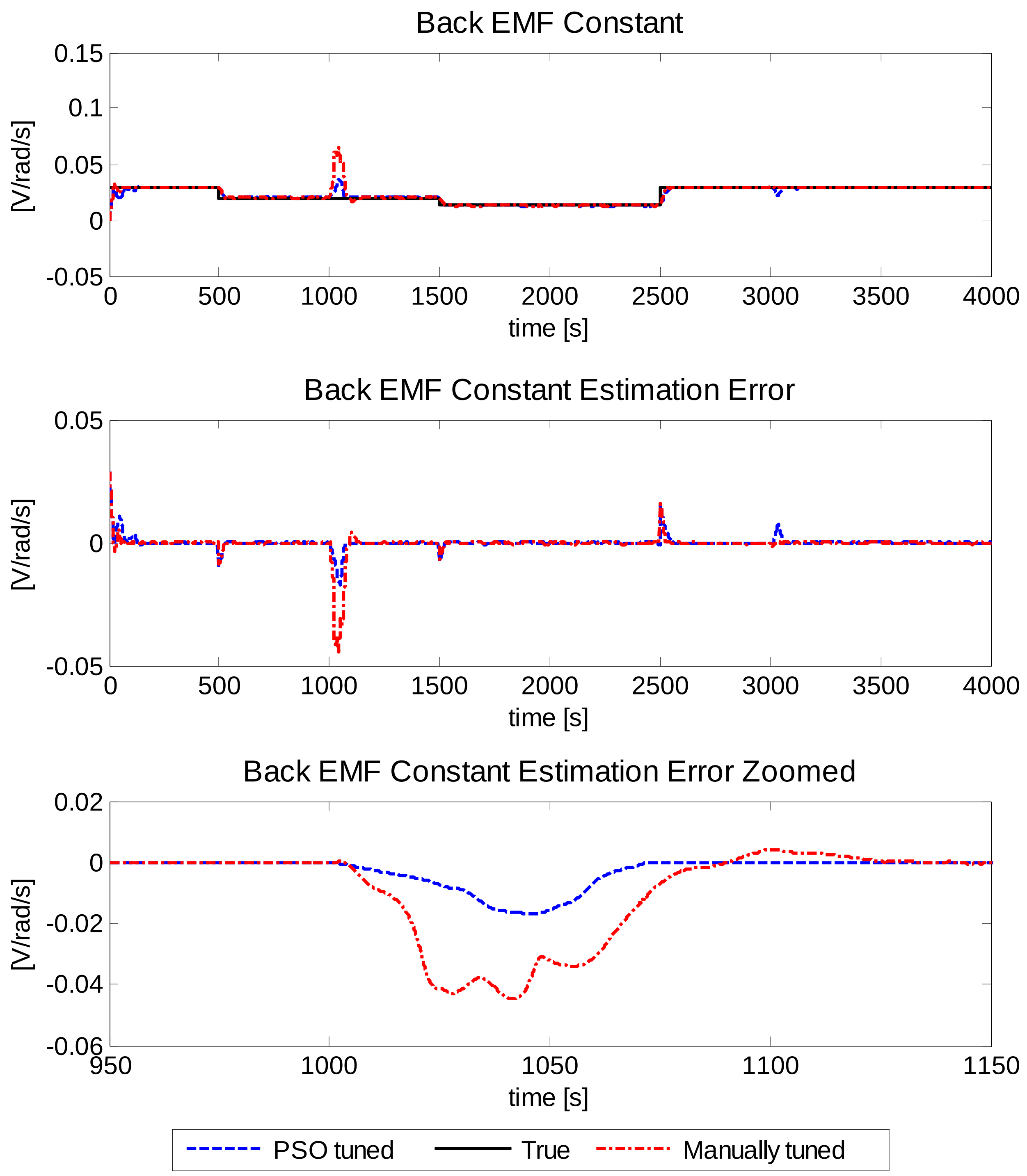

Figure 5.9 Case 2 - Medium Noise - BEMF Constant Estimates

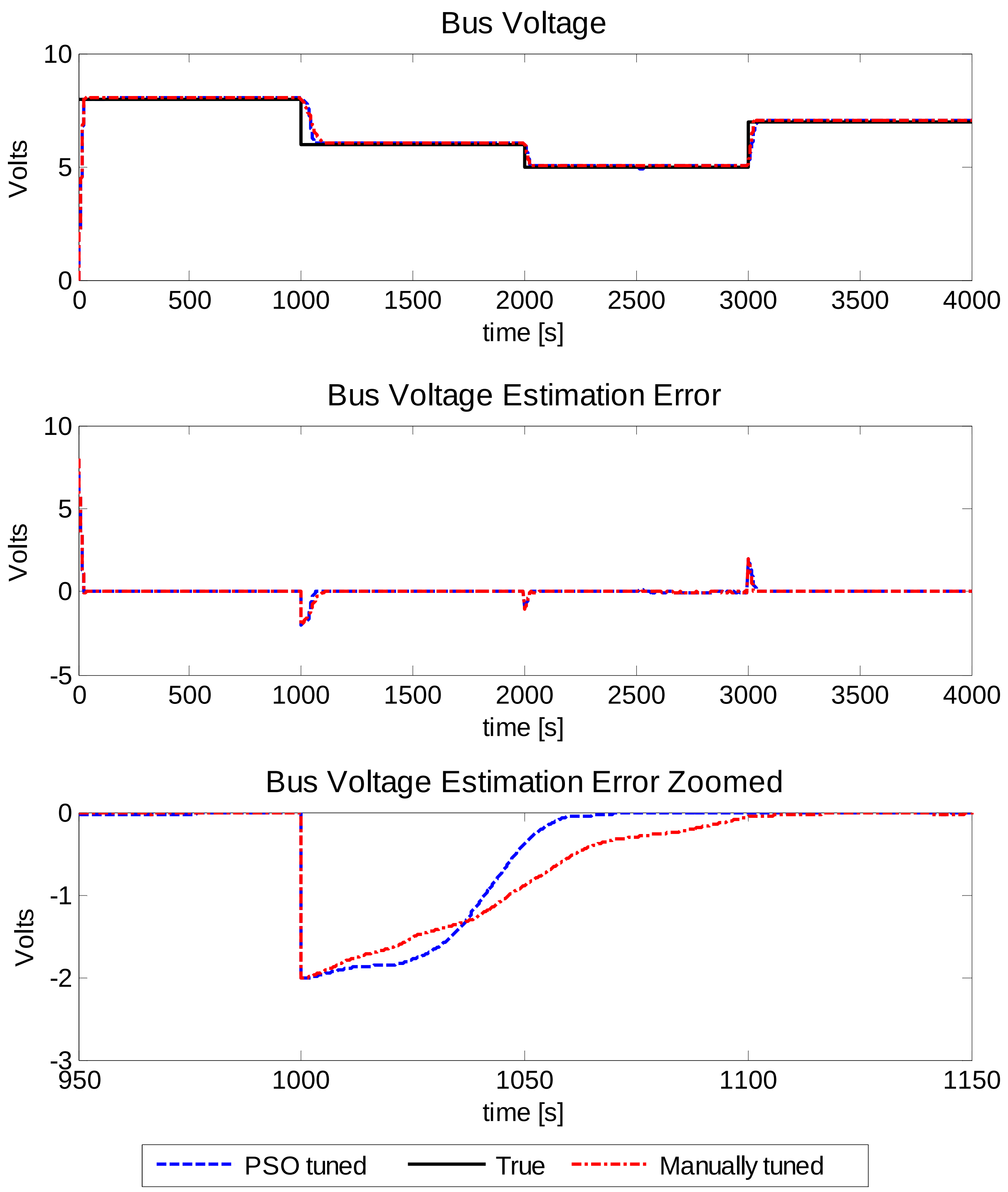

Figure 5.22 Case 2 - Medium Noise - Bus Voltage Estimates

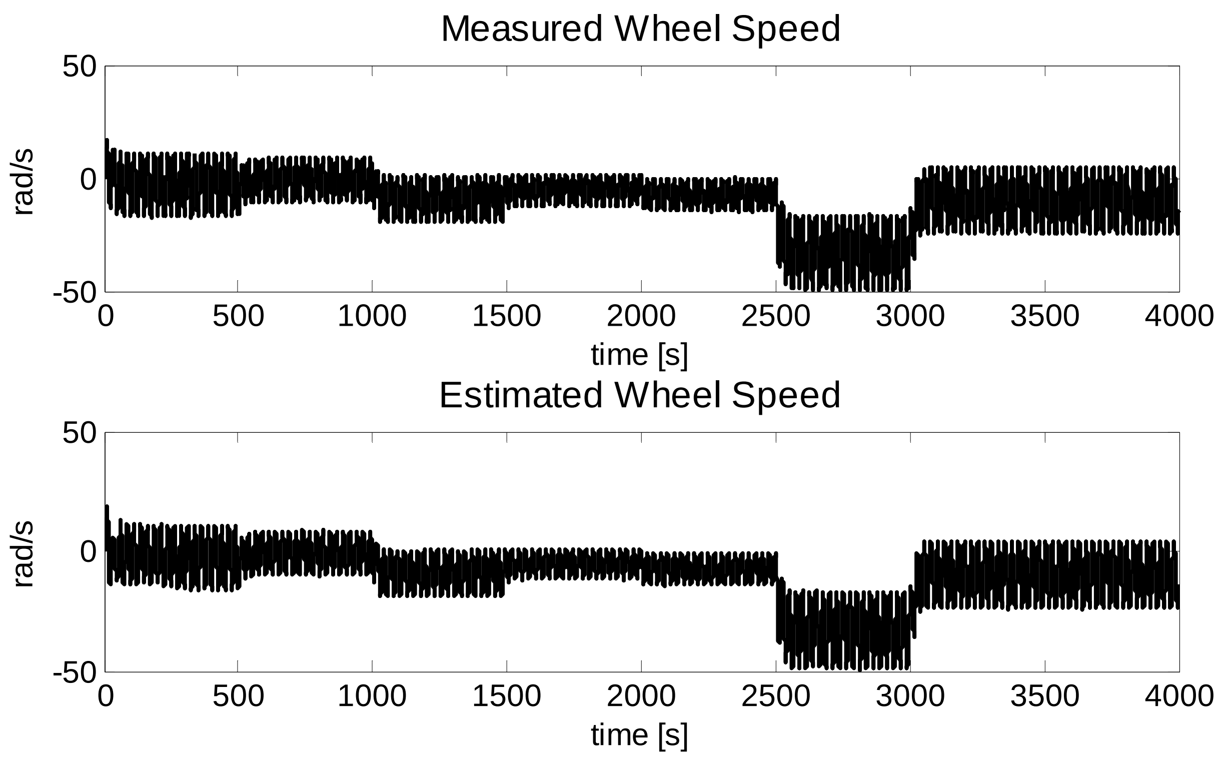

Figure 5.23 Case 2 - High Noise - Wheel Speed Measurements vs. PSO Estimates

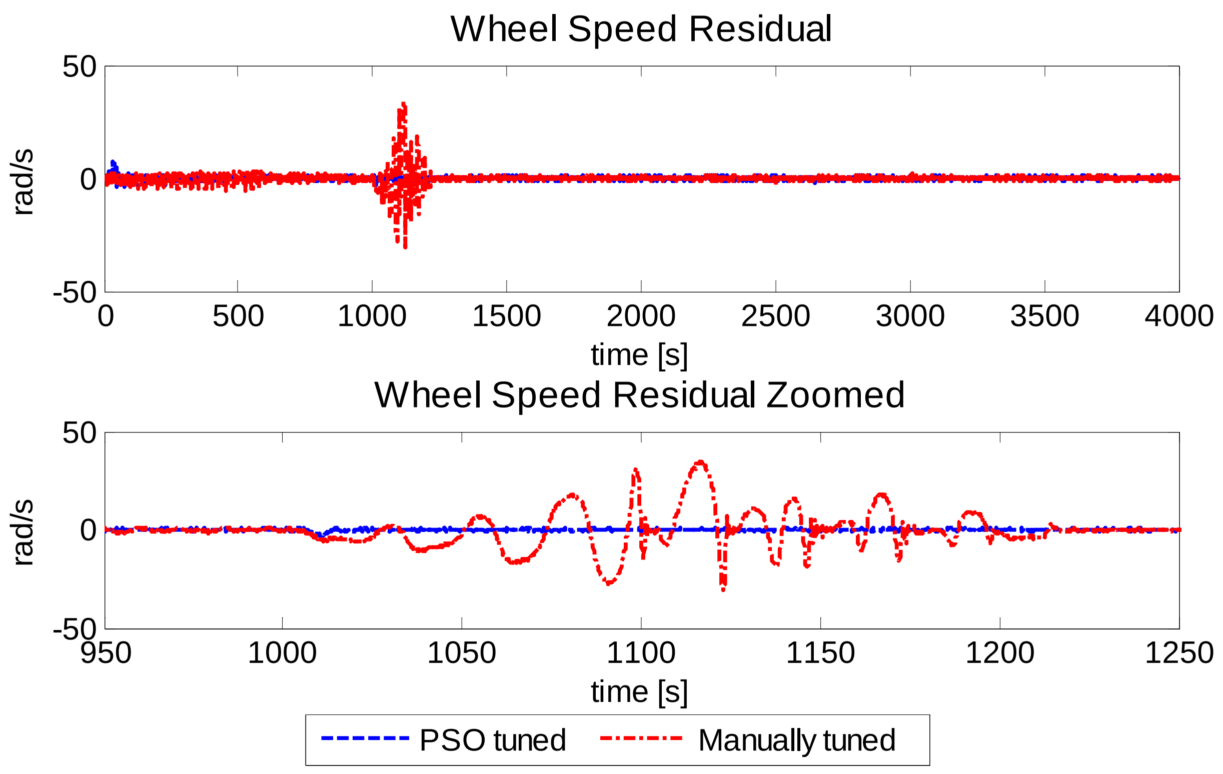

Figure 5.24 Case 2 - High Noise - Wheel Speed Residual Comparison

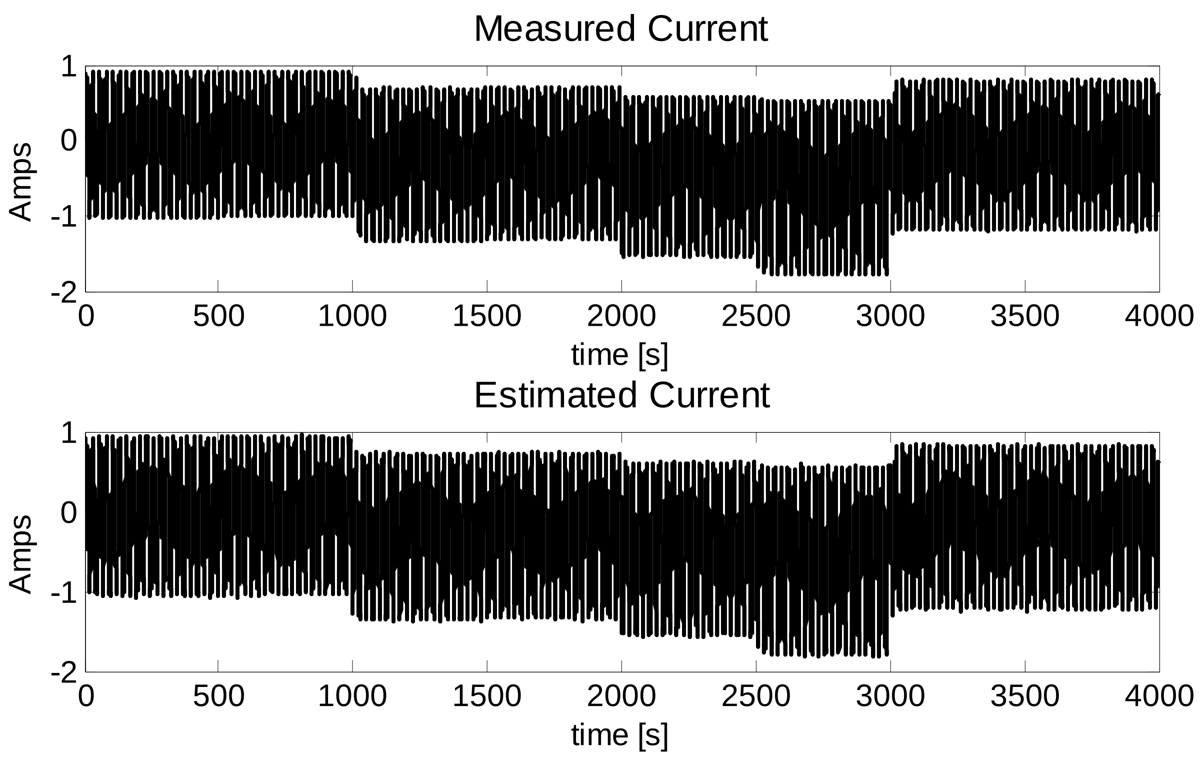

Figure 5.25 Case 2 - High Noise - Current Measurements vs. PSO Estimates

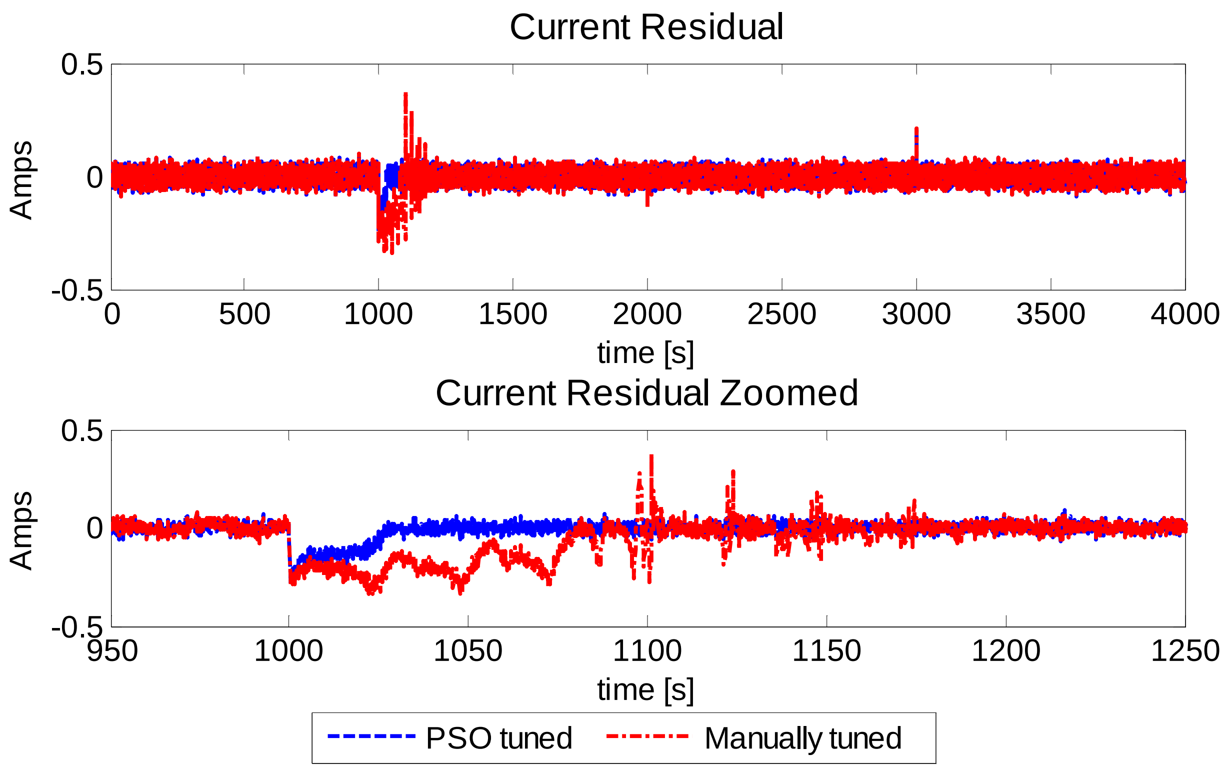

Figure 5.26 Case 2 - High Noise - Current Residual Comparison

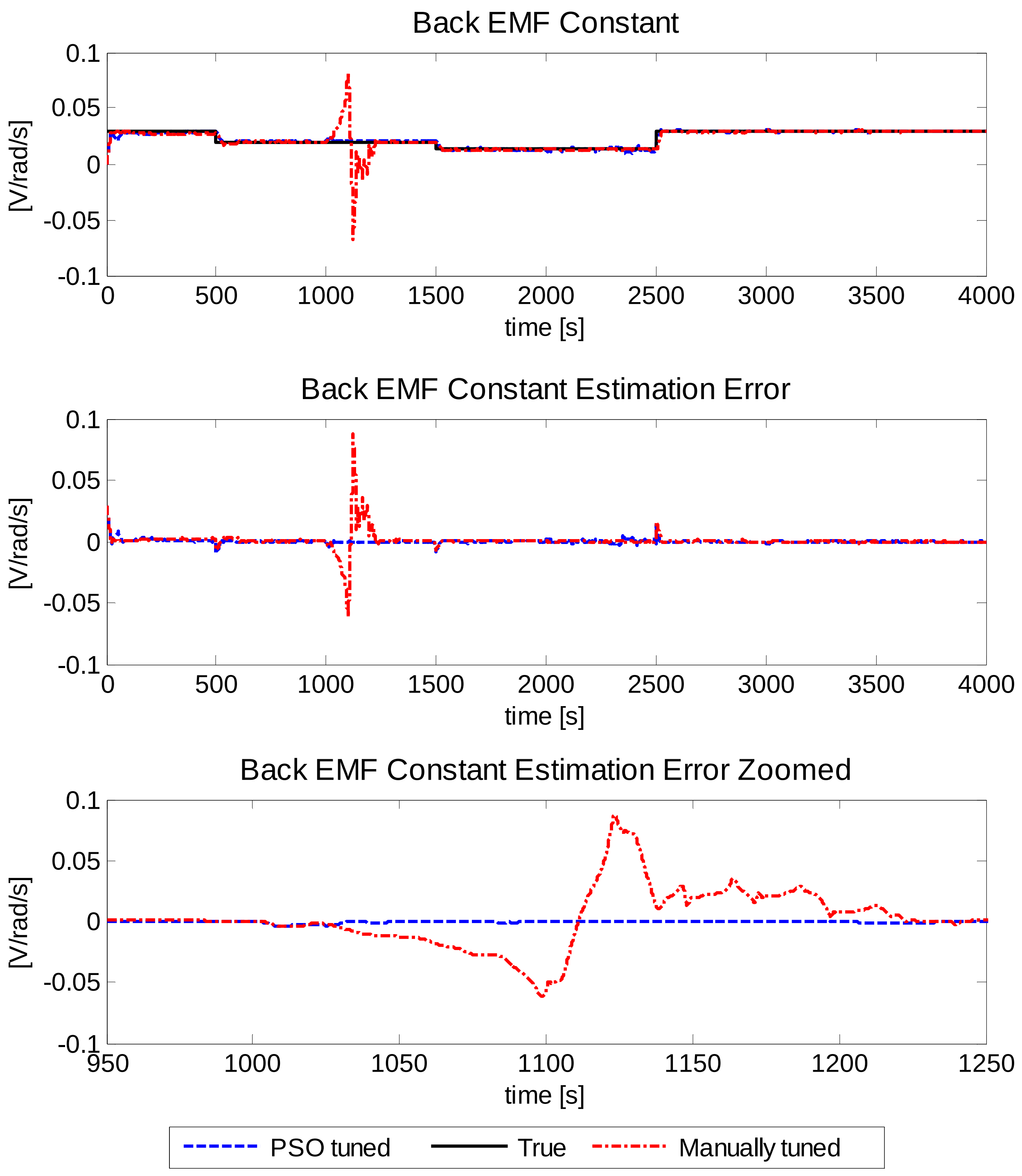

Figure 5.27 Case 2 - High Noise - BEMF Constant Estimates

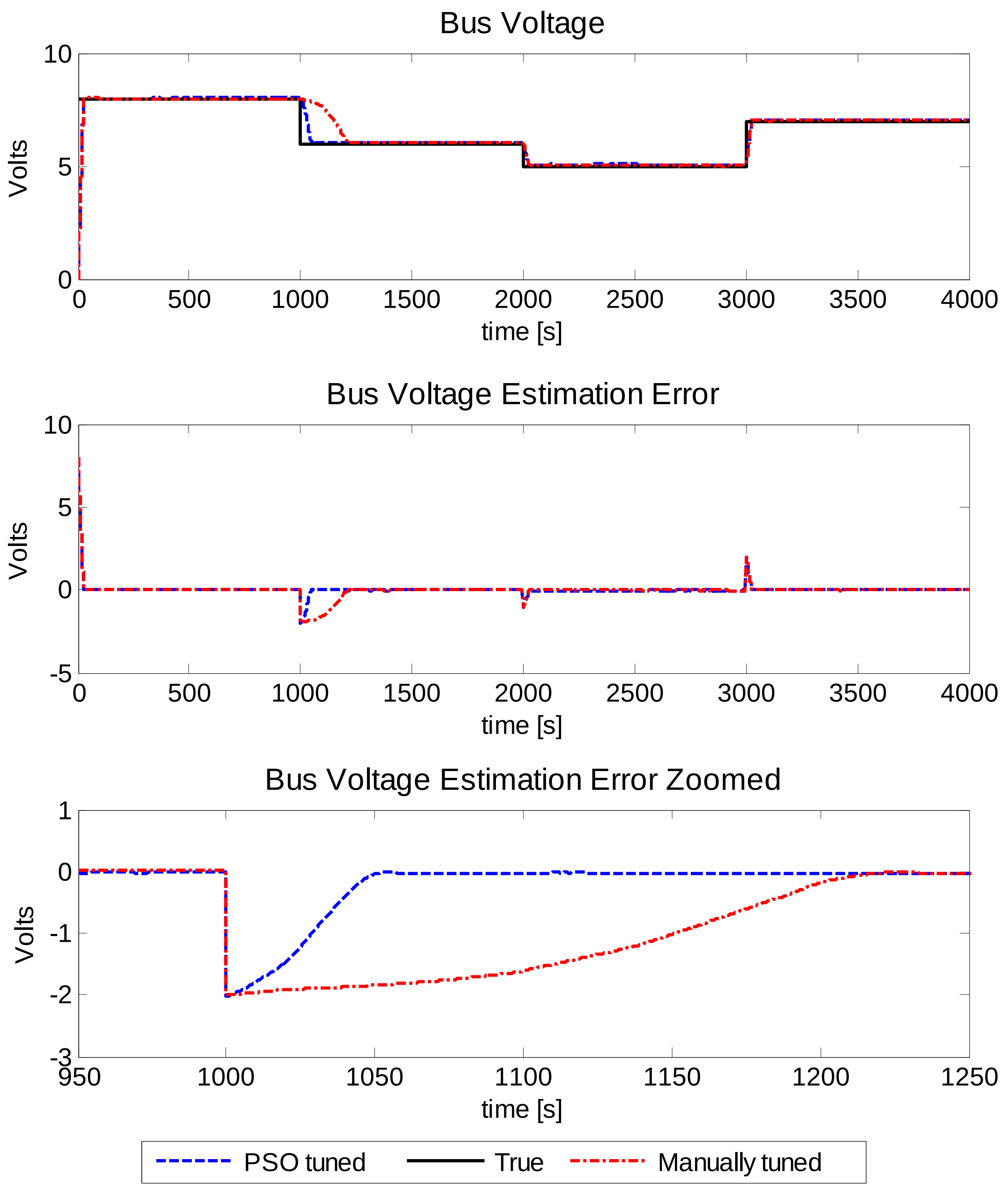

Figure 5.28 Case 2 - High Noise - Bus Voltage Estimates

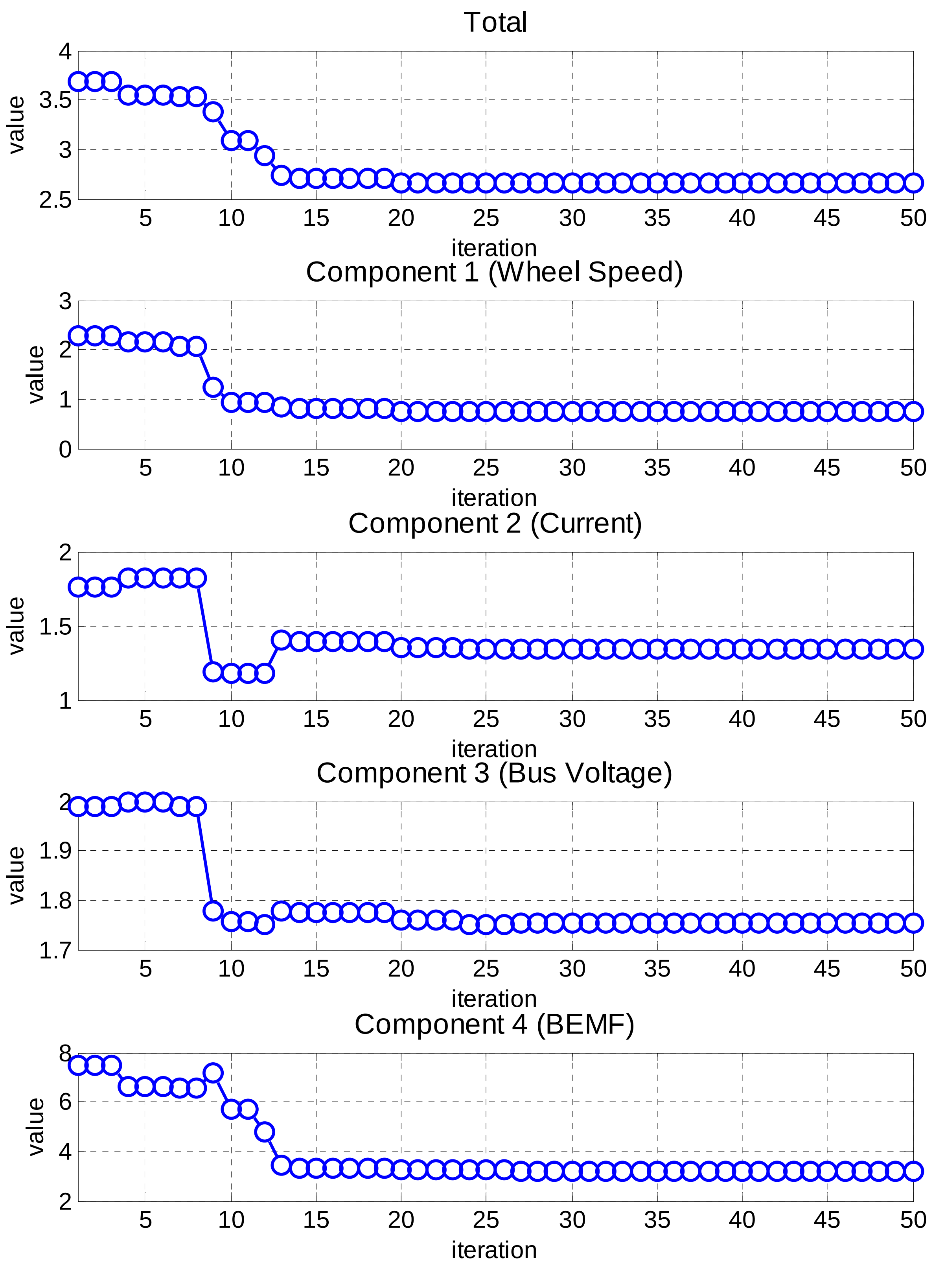

Figure 5.29 Case 2 - Objective Function Trend with Different Components Breakdown

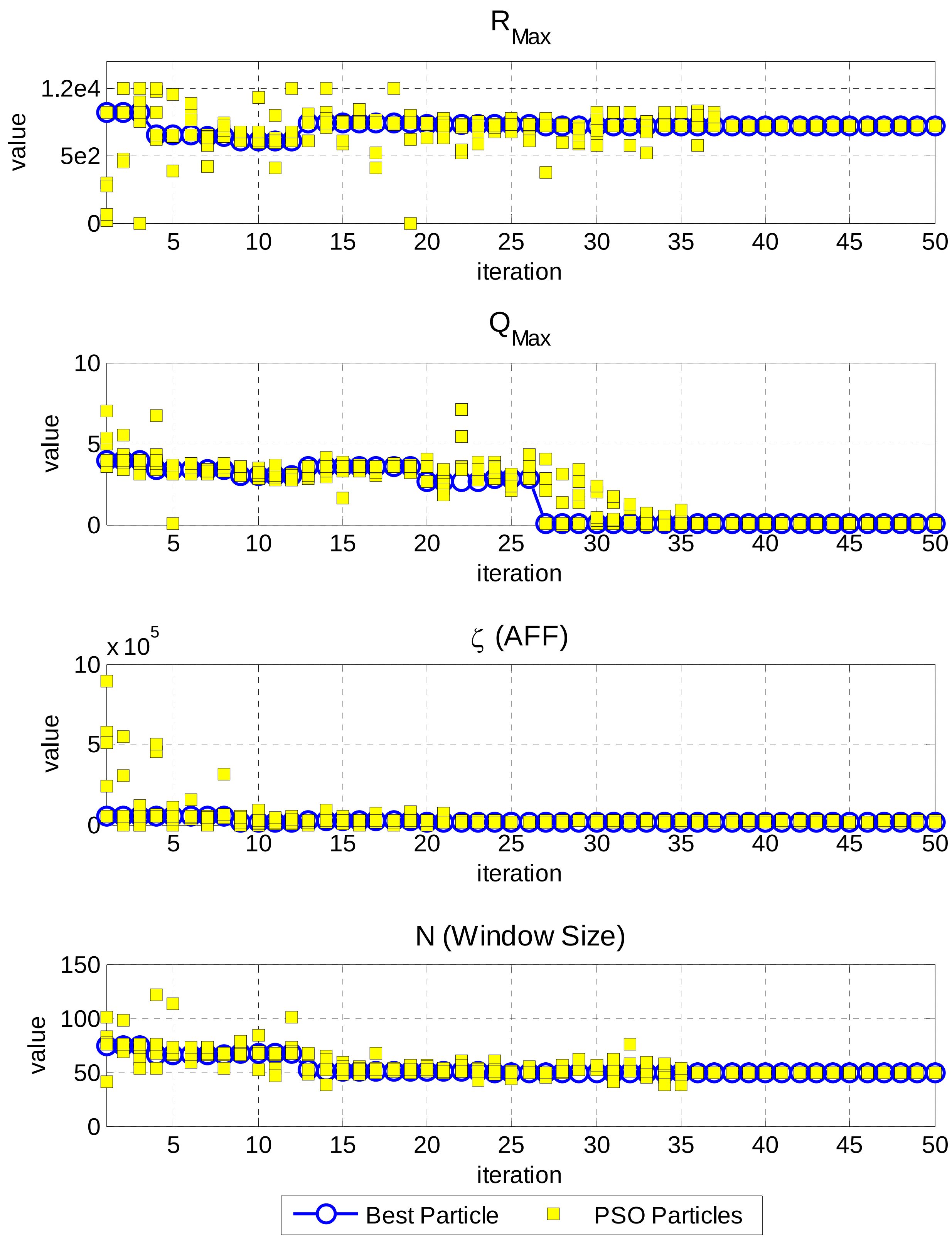

Figure 5.30 Case 2 - Parameters Convergence to the Global Optimum Trend over Iterations

Table 5.10 Case 2- Optimization Results

	Medium	High
Parameter	**Value**	
Execution time	23.63 hrs	
$R_{\max}$	7198.2	
$Q_{\max}$	0.0443	
ξ (AFF)	18261	
N $(Window\ size)$	49.98	
Total OFCPI	3.1899	2.6556
Wheel Speed OFCPI	1.8100	0.7446
Current OFCPI	0.6022	1.3464
BEMF OFCPI	1.9163	1.7528
Bus Voltage OFCPI	5.8469	3.2083

One other thing to mention is that although the values for parameters are not that close in Case 1 and Case 2 but the Total OFCPI for both is close to one another. The reason is that, as explained before, here the algorithm and consequently the system is working with a set of parameters and these parameters have influence on each other and the overall performance of the filter. Hence, there might be different sets of gains with similar or close enough performance indices. Then the choice on which one to use come the customer or the engineer who is in charge of the project.

Table 5.11 Effect of Number of Particles on the Performance of the Algorithm

		PSO				**Manual**	
		Case 2		**Case 1**			
		Medium	**High**	**Medium**	**High**	**Medium**	**High**
	Parameter	**Value**					
	Particles	5		10		N/A	N/A
	Iteration	50		40			
	Execution time	23.63 hrs		19.34 hrs		N/A	N/A
	$R_{\max}$	7198.2		1897.1		50	5
	$Q_{\max}$	0.0443		0.2565		0.01	0.005
	ξ (AFF)	18261		52765		1000	100000
	N $(Window\ size)$	49.98		164		75	200
OFPCI	Total	3.1899	3.1899	3.0686	3.0572	6.1050	8.2699
	Wheel Speed	1.8100	1.8100	1.2309	1.9821	15.3411	14.7192
	Current	0.6022	0.6022	0.8289	1.2689	0.5182	2.8936
	BEMF	1.9163	1.9163	1.9239	1.9308	19.5326	48.1118
	Bus Voltage	5.8469	5.8469	5.4327	4.1648	1.8794	2.6667

5.3 Performance Evaluation for Extreme Noise Level

Another interesting investigation was to test the performance of the algorithm on a new higher level of the noise introduced to the system. Apparently, the problem with the filter was that it could not handle high levels of noise and the estimations would vary a lot from the actual values as the noise level increased. Hence, the simulation setup was changed in a way that a higher level of noise was introduced to the system. The parameters for this simulation are listed in Table 5.12.

Table 5.12 Simulation Parameters for Extreme Noise RW FDI

Parameter	Value
Wheel Speed Noise Power	$-9.99\ dB$
Current Noise Power	$-33.01\ dB$
Sampling Period (T_s)	$0.01\ \text{s}$
Simulation Time	$4000\ \text{s}$
Window Size (N)	*
ζ	*
$\mathbf{R}_0$	$2\times10^{-4}\times I_{2\times2}$
$\mathbf{Q}_0$	$1\times10^{-4}\times I_{2\times2}$
$\mathbf{P}_0$	$1\times10^{-8}\times I_{2\times2}$
$\mathbf{Q}_{max}$	*
$\mathbf{R}_{max}$	*
$\hat{\mathbf{x}}_0$	$[8,0.029]^T$
κ	-1
n	2

5.3.1 Case 3 – 15 Iterations and 30 Particles

In order to check if the performance will be improved by increasing the number of particles as already discussed, this case is investigating the performance of the algorithm with 15 iteration and 30 particles. The search-space is the same as Table 5.8. Results for parameter estimation are shown in Figure 5.31 to Figure 5.36. The objective function trend and each individual component of the objective function trends are shown in Figure 5.37. In addition, the convergence trend for the optimization variables is depicted in Figure 5.38. Results for the parameters as well as more complementary information for the simulation are given in Table 5.13.

From the parameter estimation figures, it can be concluded that the performance has tremendously improved because the spikes in the residuals are almost gone and the estimations track the actual parameters smoother and faster as compared to the manually tuned filter. The solutions found by this methodology are near optimal and not optimal because, as explained before, and also is mentioned in Ref. [16], these types of optimizations algorithm do not guarantee to give the global optimal solution and they only provide you with near optimal solutions. Reason being that they only search an area for the solution and the global optimal solution may not lie within that region in the space. However, they still give a decent estimate of were the global optimal could be or where you can search further for the optimal solution. This is why many of these evolutionary swarm-based algorithms are linked with gradient-based algorithms known as “hybrid approaches” to give better results than the one each could individually give to the user.

One thing to note here is that the results for this simulation are not shown for the converged solution with all particles settled in one position. The reason is that the machine that the simulations were executed on was not powerful enough and the memory that was used was overflown after certain number of iterations. Convergence is achieved when after certain number of iterations, there is no change in the objective function value and there is no change in the position of the particles in the search-space, which means all particles have converged to a solution for the problem. However, the tolerance between these final iterations could vary depending on the designer of the algorithm but for an ideal case this tolerance should be set to zero.

As can be seen from Figure 5.38, not all particles have converged to the solution and the reader needs to remember that these figures do not show the final converged solution for this case and more iterations are required for the simulation to guarantee the solution has converged and the solution is a near-optimal solution for the problem. In comparison with the results for the medium and high noise levels, the estimations for this case are noisier and less accurate. This is because of the noise level and the sensitivity of the filter to the noise level. One needs to consider that the performance of the filter in this case is under the most severe noise that the system could sustain and in practice, the system would fail. However, as mentioned earlier, this case was studied to show the superiority of the proposed methodology over existing trial and error procedures.

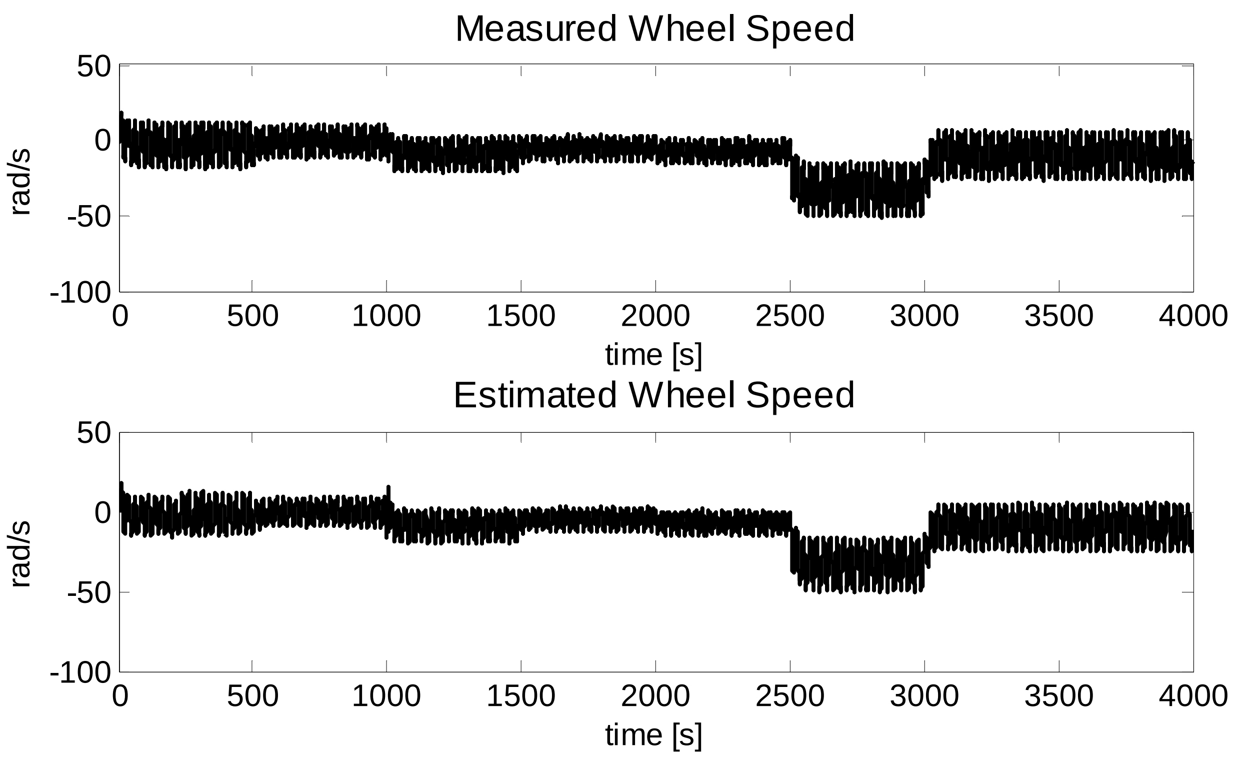

Figure 5.31 Case 3 - Extreme Noise - Wheel Speed Measurements vs. PSO Estimates

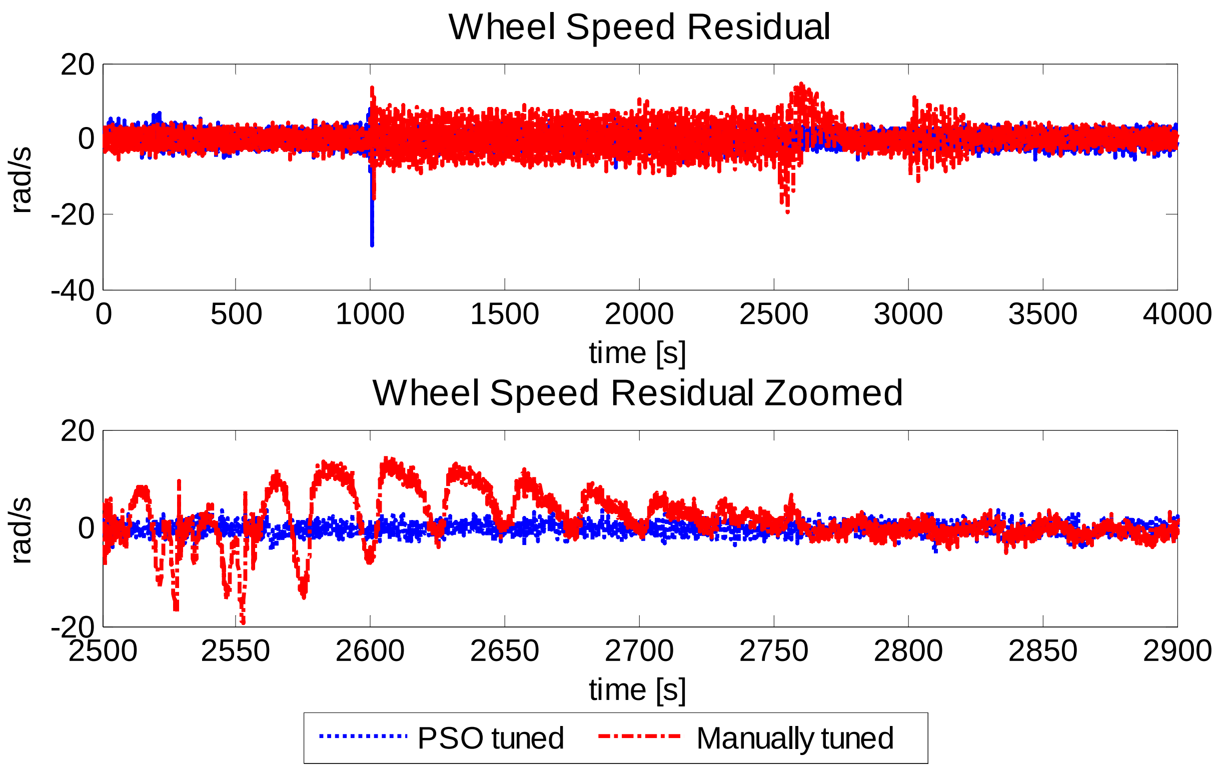

Figure 5.32 Case 3 - Extreme Noise - Wheel Speed Residual Comparison

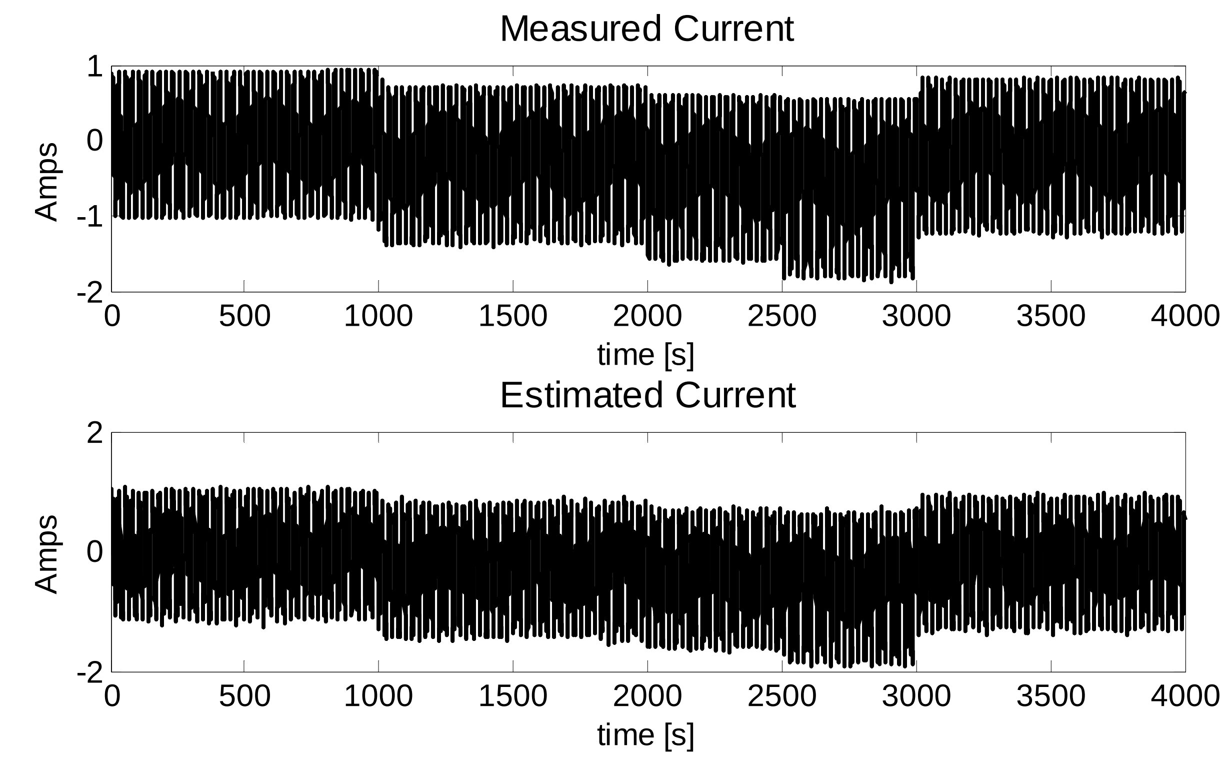

Figure 5.33 Case 3 - Extreme Noise - Current Measurements vs. PSO Estimates

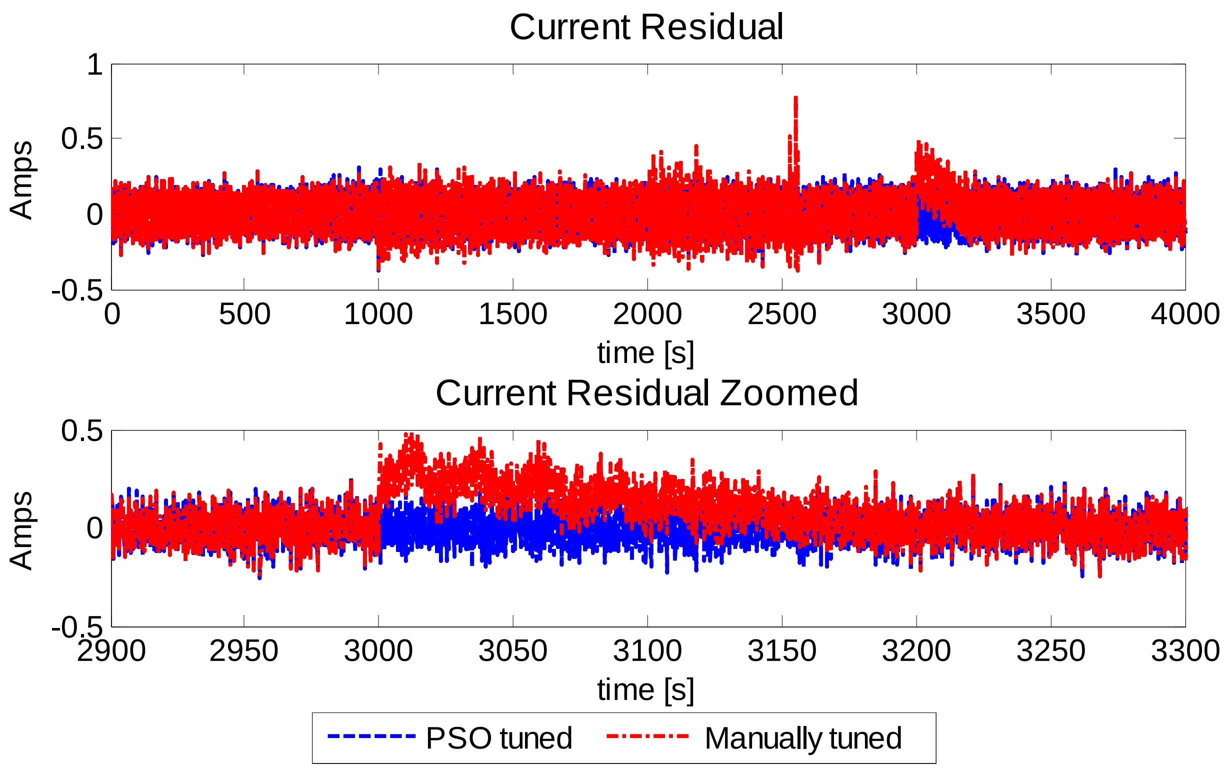

Figure 5.34 Case 3 - Extreme Noise - Current Residual Comparison

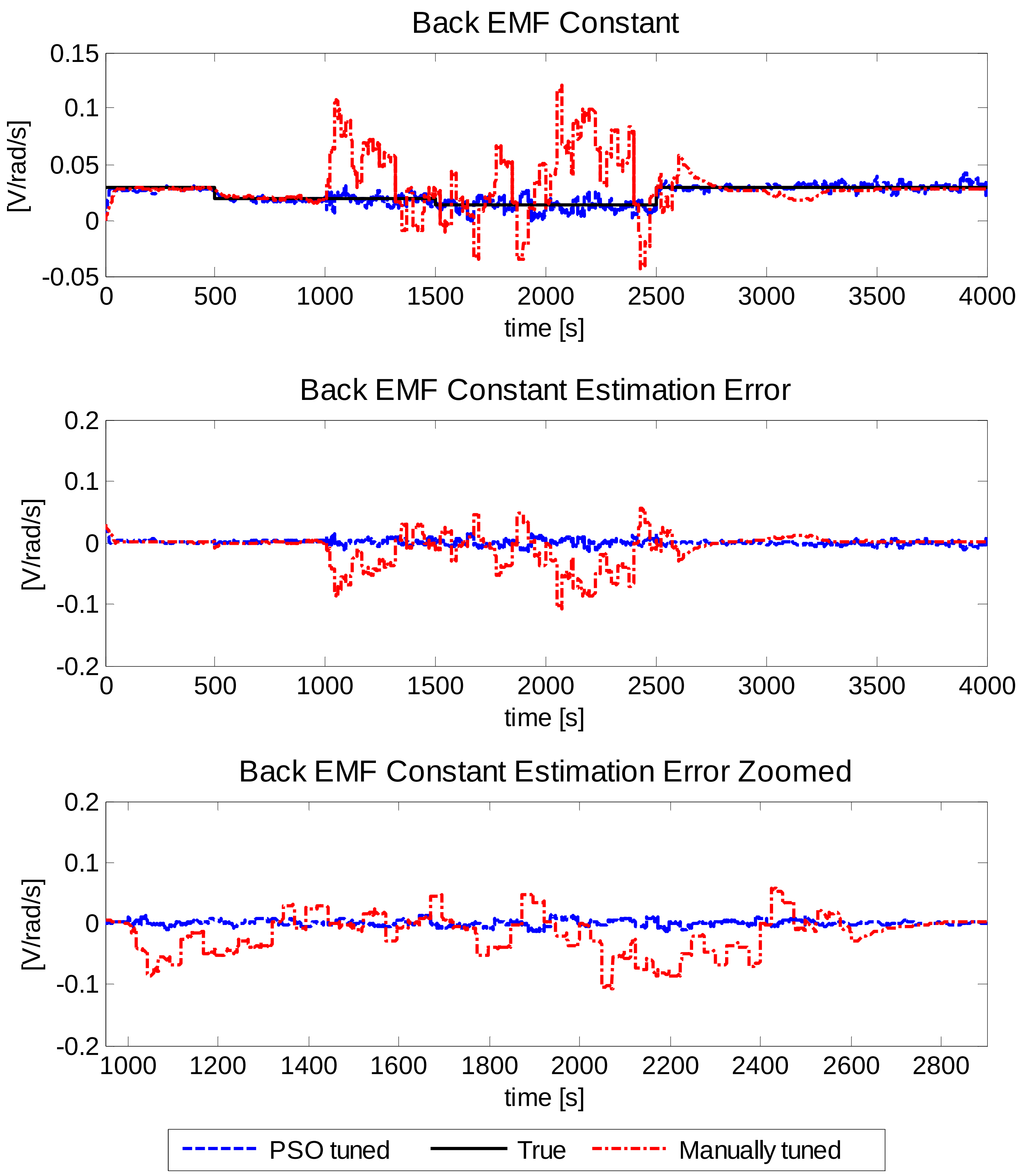

Figure 5.35 Case 3 - Extreme Noise - BEMF Constant Estimates

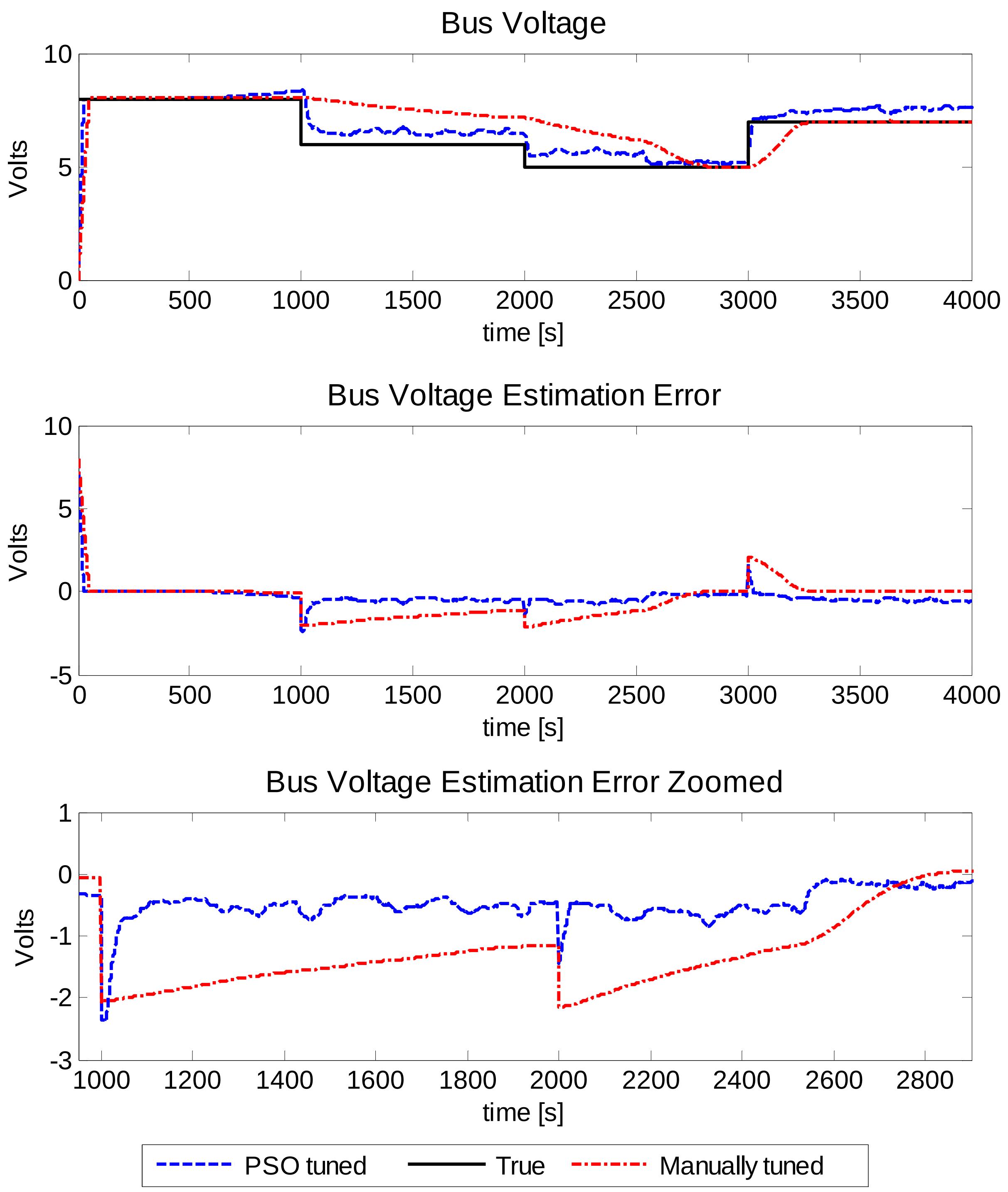

Figure 5.36 Case 3 - Extreme Noise - Bus Voltage Estimates

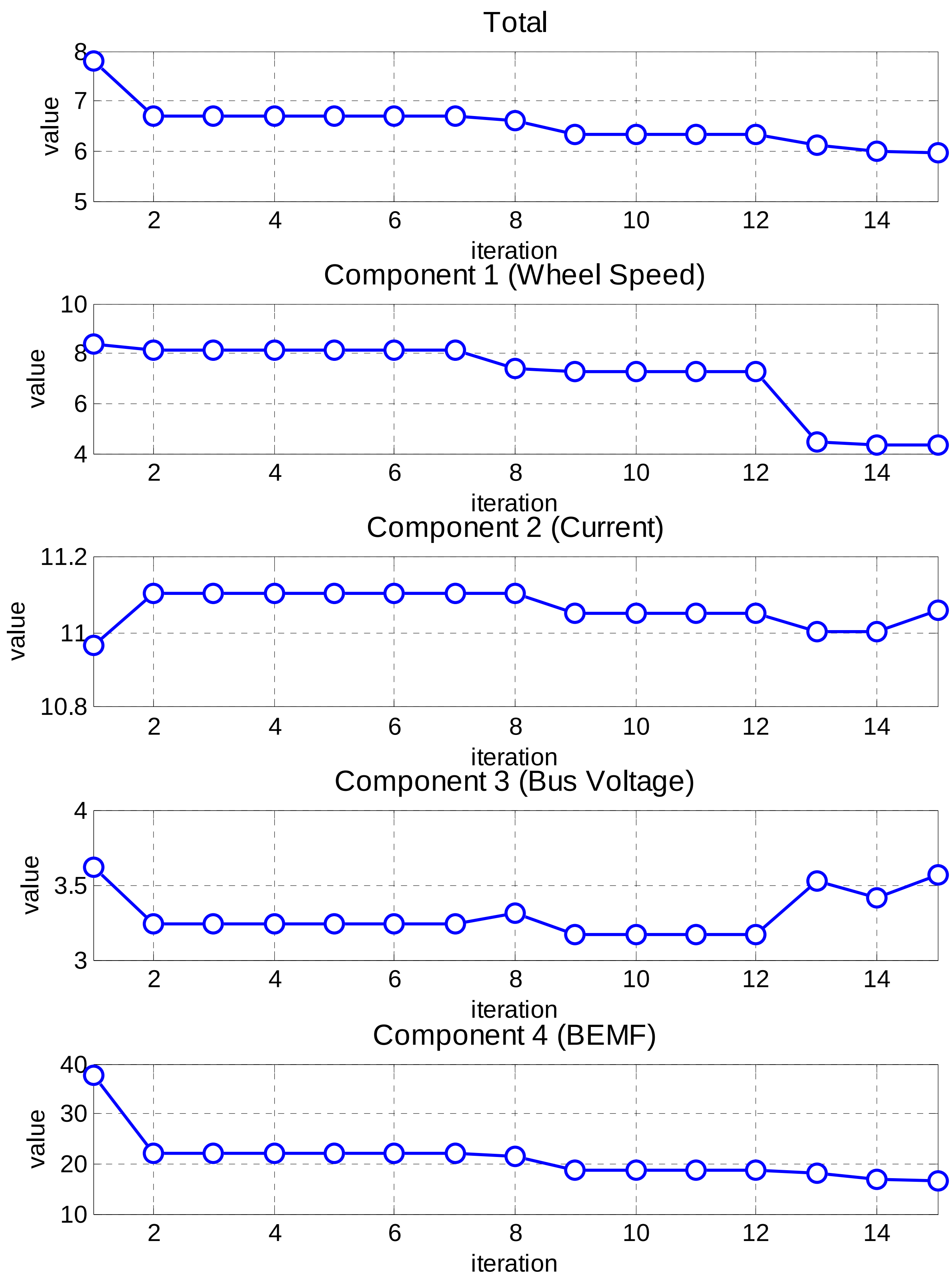

Figure 5.37 Case 3 - Objective Function Trend with Different Components Breakdown

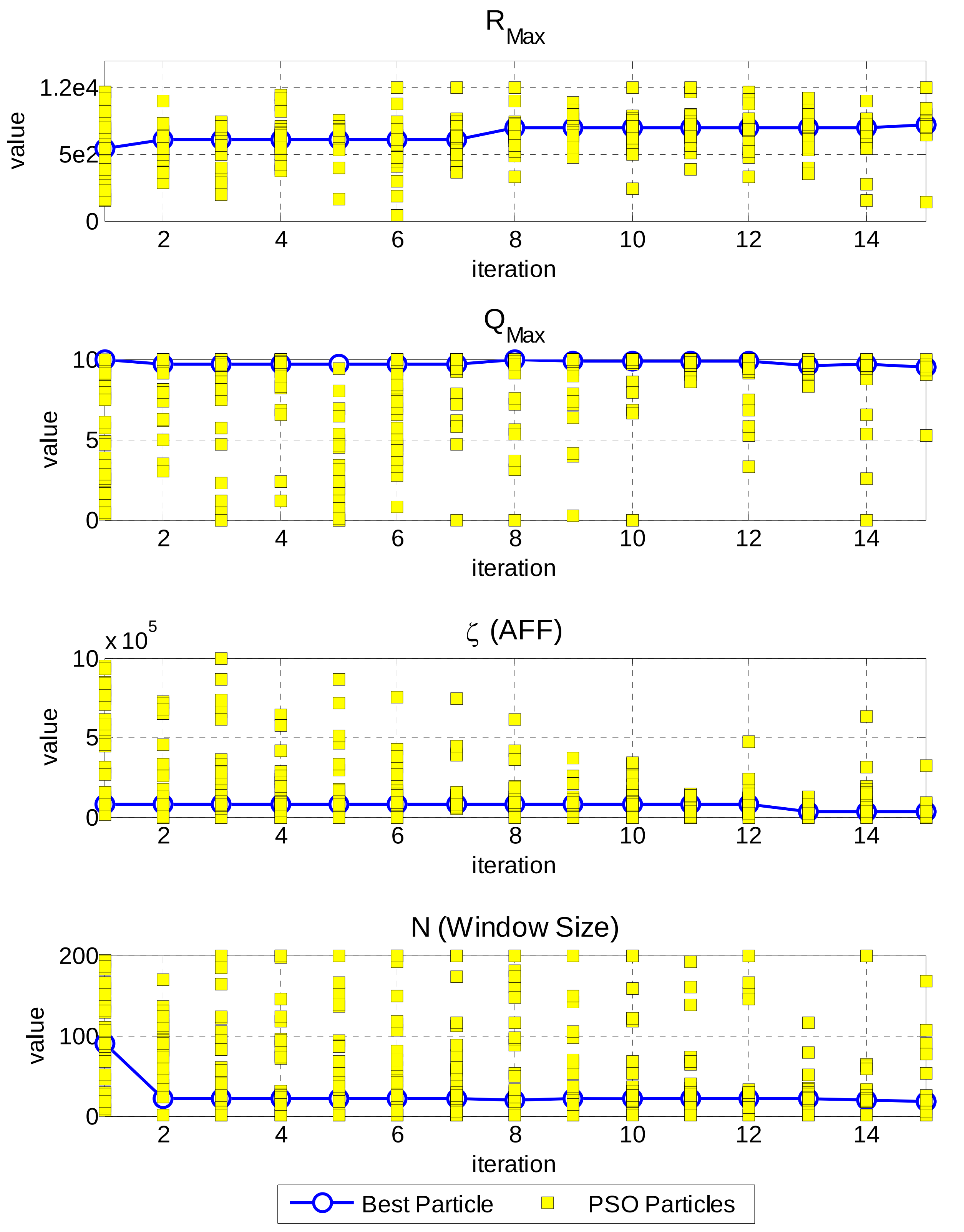

Figure 5.38 Case 3 - Parameters Convergence to the Global Optimum Trend over Iterations

Table 5.13 Case 3- Optimization Results

Parameter	**Value**
Execution time	18.2 hrs
$R_{\max}$	7166
$Q_{\max}$	9.5273
ξ (AFF)	37296
N $(Window\ size)$	18
Total OFCPI	5.9710
Wheel Speed OFCPI	4.3546
Current OFCPI	11.0594
BEMF OFCPI	16.6727
Bus Voltage OFCPI	3.5659

Overall, the results in all the cases investigated above, show that the performance of the filter has tremendously improved after being automatically tuned with the proposed PSO-based methodology in this thesis. In this case, the investigation proved that for the systems and noise levels that the algorithm has no knowledge about, it could still perform pretty well and give satisfactory sets of gains/parameters for the filter to minimize the residuals and lessen the tracking delay in the filter.

5.4 Conclusions

The proposed tuning methodology based on Particle Swarm Optimization explained in Chapter 4, has proven to perform well for a wide range of noise levels in the system. It also shows that without any previous knowledge of the system the methodology can still perform well and tune the filter in less time and with better performance indices.

The proposed methodology can handle **Concurrent Optimization for Different Noise Levels;** and with its **Modular Structure,** it can be used for different purposes and be modified by different people without having them sit and discuss the results together. The **Adaptability** of the methodology makes it easy to use and implement into different systems with minimum adjustments. It also is designed on a **Plug & Play** policy making it extremely easy to plug a system (Simulink model) to the algorithm without having trouble changing any part of the algorithm and then initialize the parameters for the system and let it just work. It is also capable of **Handling Different Types of Variables**, which gives designers more freedom on which types of variable they can use. The robustness of the proposed methodology to the noise level is another important feature of it. It can handle **Extreme Noise Levels** and with that, it can be claimed that any other sort of disturbance or change in the system can be compensated by the algorithm with a new set of gains.

In the next Chapter, more explanations are given on each of the above-mentioned advantages of the proposed algorithm over the existing methods in the literature. In fact, the proposed methodology is combining different sciences and technologies making it feasible for it to be adaptable to different systems while maintaining the performance satisfactory.

6. Conclusions and Future Work

This thesis approached the problem of tuning a filter/controller properly with the minimum human interference involved. The idea was inspired by the lack of such systematic tuning methodology in the SSDC lab. As explained in the introduction section. The idea was to develop a methodology, which could do this task automatically and with the minimum human interference possible. The methodology needed to be independent of the system it was being applied to and needed to be adaptable and easy to use for different systems. It needed to have capability of handling different noise levels, being able to have flexible data type and input/output setting and a search-space so that the implementation of the system into other systems could be possible, efficient and easy to work with. In this thesis, a methodology was proposed to do the task. Systematic tuning of a filter/controller was proposed and the verification of the performance for the proposed methodology was tested for tuning an Adaptive Unscented Kalman Filter (AUKF) for used for FDI purposes on a Reaction Wheel (RW) unit. There were four parameters that needed tuning in this specific problem and the case studies were presented to investigate different parameters that affected the performance of the algorithm such as number of iterations, number of particles, both of the previous parameters concurrently and also the system parameters, dynamics and disturbances such as noise level, and influence of each on the performance. The overall conclusion on the performance of the filter is that it proved to perform well in the most severe case of a faulty system with high noise level as compared to the manually tuned filter in Ref. [16]. It also showed that with a maximum noise level that could be introduced to the system the algorithm still gave satisfactory results proving that it is robust to changes in the system as well as noise level. It is also worthy to mention that in order to present the work there was a need for an intuitive representation of the convergence of the particles in a multi-dimensional space which in this thesis was done by the author

6.1 Summary of Contributions

As mentioned earlier the main focus of this thesis was to develop a methodology to tune any type of filter/controller with the minimum human interference involved. To do so a methodology was proposed and tested for verification purposes on the performance with a case study of a FDI problem for a RW unit. The following features are the main contributions of this work in the proposed methodology.

Concurrent Optimization for Different Noise Levels: This methodology is able to handle different noise levels at the same time and give set of gains/parameters for the filter/controller that can be used for all noise levels. The way the methodology handles the multi-level noise is by running the simulation with the parameters fed into the system from the optimization algorithm and check if the simulation converges for one noise level then it will go to the next noise level simulation and it will continue doing this until the last noise level simulation is converged. If in any of the noise levels the simulation diverges, then the methodology will consider that position or set as a useless point and the search will continue for the solution away from that point. In terms of how the objective function will handle the multi-level noise situation, it comes to the designer because in this thesis the focus was on higher levels of noise but as explained in the previous section if the objective function is constructed in a way to account for all the parameters in all noise levels then it can output results which will guarantee satisfactory results for all noise levels and all parameters included in the objective function.

Modular Structure: The proposed methodology has modular structure, meaning that it can be used for different purposes and be modified by different people without having them sit and discuss the results together. The modules are input/output base. This means that each module inputs some parameters and outputs some other parameters and the designers need to know only those inputs and outputs are handled correctly. Other than that, whatever is going on within each module is dependent on the designer and could, in some cases, be considered as black-box policy.

Adaptability: Different systems have different parameters and dynamics and for different filters/controller, there is a requirement for different unrelated parameters to be tuned. The proposed methodology is easy enough to use and can be applied to different systems with minimum requirement on adaptation and/or adjustments. Parameters for tuning can be defined easily and the search-space could be set in just changing few lines of code. The rest is in control of the methodology and it will run the simulation and output required parameters without knowing anything about the system.

Plug & Play: One of the most important things about different methodologies is how easily they can be applied and used. With the modular structure of the proposed methodology it is extremely easy to plug a system (Simulink model) to the algorithm without having trouble changing any part of the either and then initialize the parameters for the system in the initialize block and then set the parameters that you want to tune with the desired search-space and the algorithm just works. This becomes more important when industries have models of a system that they do not want the outside engineers know about (known as “Black Box Policy”) and this way the tuning process can be done confidential with the procedure explained above. This helps industry and universities to collaborate more in industrial/scientific projects.

Handling Different Types of Variables: As explained in the previous sections. This methodology can handle different variable types as well. In this thesis, there was windows size parameter with integer type and there were other parameters with floating type. This gives designers more freedom on which types of variable they can use during the process of design so that in the tuning part they would not face any issues.

Performance for the Extreme Noise Level: In the last case series presented in this thesis, it was showed that the methodology is capable of handling Extreme Noise level in the system, which was not done before in the work that this study was based on for the comparison purposes. It can be claimed that the noise level applied to the system was at its extremes because as listed in the tables for each simulation it can be seen that the noise levels are at their extremes. This means that if this methodology is applied to other systems with higher noise levels it still is capable of giving good results and finding near optimal solutions for the problem in question.

6.2 Future Work

For the future work on the existing work presented in this thesis, there are some suggestions as follows:

Implementing the Methodology on Other Systems: In order to investigate the performance of the methodology on other systems as well as the applicability of the implementation, one needs to implement the methodology and evaluate the easiness of the implementation as well as the performance of the proposed methodology. This could be done on any other system but the suggestion is to start with systems having simpler dynamics and then if further investigations are required expand it to systems that are more complex.

Implementing the Methodology on Other Controllers: In this thesis, the methodology was applied to an AUKF filter and it was mentioned in the contents that this algorithm could also be used to tune controllers as well as filters. In order to investigate the performance of the algorithm on other controllers/filters, one needs to do the implementation and investigate the applicability, efficiency, accuracy and overall performance of the methodology.

Hybrid Approach: As mentioned in the text, the methodology proposed here is computationally heavy, meaning that it requires a lot evaluations of the objective function (which necessarily requires execution of the simulation mode) and it could take long times. In order to improve the execution time, this algorithm can be mixed with a gradient-base method to further improve the results as well as decrease the execution time.

Test on Super Computers: As far as the author knows, there is a computing centre in Canada that provides access to super-fast computers for experimental, educational and research purposes to all researches and people in the field. One interesting investigation would be testing the performance of the algorithm (execution time specifically) on these super-fast computers provided by High Performance Computing Virtual Laboratory (HPCVL) and propose this approach and tool to be used for future implementations of the methodology so that the execution time will be tremendously decreased (as the author believes so).

7. References

[1] Mahoney, M.S., and Heilbron, J., "The mathematical career of Pierre de Fermat (1601–1665)," *History: Reviews of New Books,* Vol. 1, No. 8, 1973, pp. 180-180.

[2] Rockafellar, R.T., "Lagrange multipliers and optimality," *SIAM Review,* 1993, pp. 183-238.

[3] Fletcher, R., "Practical methods of optimization, Volume 1," Wiley, 1987,

[4] Dantzig, G.B., "Linear programming and extensions," Princeton University Press, 1998, pp. 648.

[5] Kantorovich, L.V., "The mathematical method of production planning and organization," *Management Science,* Vol. 6, 1939, pp. 363-422.

[6] Åkesson, B.M., Jørgensen, J.B., Poulsen, N.K., "A tool for kalman filter tuning," *Computer Aided Chemical Engineering,* Vol. 24, 2007, pp. 859-864.

[7] Schwaab, M., Biscaia, E.C., Monteiro, J.L., "Nonlinear parameter estimation through particle swarm optimization," *Chemical Engineering Science,* Vol. 63, No. 6, 2008, pp. 1542-1552.

[8] Massoudieh, A., Mathew, A., and Ginn, T.R., "Column and batch reactive transport experiment parameter estimation using a genetic algorithm," *Computers & Geosciences,* Vol. 34, No. 1, 2008, pp. 24-34.

[9] Kang, C.W., and Park, C.G., "Attitude estimation with accelerometers and gyros using fuzzy tuned Kalman filter," *European Control Conference,* 2009, pp. 3713-3718.

[10] Pedchote, C., and Purdy, D., "Parameter estimation of a single wheel station using hybrid differential evolution," *Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering,* Vol. 217, Sage Publications, 2003, pp. 431-447.

[11] Jatoth, R.K., and Kumar, T.K., "Particle Swarm Optimization Based Tuning of Unscented Kalman Filter for Bearings Only Tracking," *International Conference on Advances in Recent Technologies in Communication and Computing, ARTCom,* IEEE, 2009, pp. 444-448.

[12] Jatoth, R.K., and Kumar, T.K., "Swarm Intelligence Based Tuning of Unscented Kalman Filter for Bearings Only Tracking," *International Journal of Recent Trends in Engineering,* Vol. 2, No. 5, 2009, pp. 177-181.

[13] Jatoth, R.K., and Kumar, T.K., "Particle Swarm optimization Based Tuning of Extended Kalman Filter for Manoeuvring Target Tracking," *International Journal of Circuits, Systems and Signal Processing,* Vol. 3, No. 3, 2009, pp. 127-136.

[14] Xu, X., and Wang, X., "Hard-Failure Diagnosis Using Self-tuning Kalman Filter," *Intelligent Computing and Information Science,* Vol. 134, 2011, pp. 326-334.

[15] Lau, T.K., and Lin, K., "Evolutionary tuning of sigma-point Kalman filters," *IEEE International Conference on Robotics and Automation,* IEEE, 2011, pp. 771-776.

[16] Powell, T.D., "Automated tuning of an extended Kalman filter using the downhill simplex algorithm," *Journal of Guidance, Control, and Dynamics,* Vol. 25, No. 5, 2002, pp. 901-908.

[17] Abreu, N., "Fault Diagnosis With Adaptive Kalman Filters and CMG Design for Picosatellite ACS," *Master's Thesis, Department of Aerospace Engineering, Ryerson University,* 2010,

[18] Sobhani-Tehrani, E., and Khorasani, K., "Identification For Nonlinear Systems Using Hybrid Approach," *Master's Thesis,* 2008, pp. 12-13, 15, 18, 37, 92.

[19] Cheung, J.T., and Stephanopoulos, G., "Representation of process trends - Part 1. A formal representation framework," Vol. 14, No. 4/5, 1990, pp. 495-510.

[20] Myoken, H., "State-space representation of dynamical econometric model and identifiability conditions," North-Holland, 1976, pp. 635-642.

[21] Min, W., and Shiyin, Q., "Robust fault diagnosis for reaction flywheel based on reduced-order observer," *Seventh International Symposium on Instrumentation and Control Technology: Optoelectronic Technology and Instruments, Control Theory and Automation, and Space Exploration - International Society for Optical Engineering (SPIE),* Vol. 7129, 2008, pp. 71292H-1-71292H-8.

[22] Ke, Z., Bin, J., and Peng, S., "Adaptive observer-based fault diagnosis with application to satellite attitude control systems," *Second International Conference on Innovative Computing, Information and Control,* 2007, pp. 2035-2038.

[23] N. Tudoroiu, K.K., "Fault detection and diagnosis for reaction wheels of satellite's attitude control system using a bank of Kalman filters," *International Symposium on Signals,* Vol. 1, 2005, pp. 199-202.

[24] Chow, E., and Willsky, A.S., "Analytical redundancy and the design of robust failure detection systems," *IEEE Transactions on Automatic Control,* Vol. 29, No. 7, 1984, pp. 603-614.

[25] Neguang, S.K., Zhang, P., and Ding, S., "Parity based fault estimation for non-linear systems: An LMI approach," *Proceedings of American Control Conference,* Minneapolis, Minnesota, USA, 2006, pp. 5141-5146.

[26] Hofling, T., and Isermann, R., "Adaptive parity equations and advanced parameter estimation for fault detection and diagnosis," *13th World Congress, International Federation of Automatic Control. Fault Detection, Pulp and Paper, Biotechnology,* 1996, pp. 55-60.

[27] Sobhani-Tehrani, E., Talebi, H.A., and Khorasani, K., "Neural parameter estimators for hybrid fault diagnosis and estimation in nonlinear systems," *IEEE International Conference on Systems, Man and Cybernetics,* 2008, pp. 3171-3176.

[28] Xu, S., and Xiao, D., "A new fault diagnosis method based on parameter estimation," *Control Theory and Applications (Chinese),* Vol. 18, No. 4, 2001, pp. 493-497.

[29] Boskovic, J.D., Li, S.M., and Mehra, R.K., "Intelligent control of spacecraft in the presence of actuator failures," *38th IEEE Conference on Decision and Control,* Vol. 5, 1999, pp. 4472-4477.

[30] Wu, Q., and Saif, M., "Robust fault diagnosis for a satellite large angle attitude system using an iterative neuron PID (INPID) observer," *American Control Conference,* 2006, pp. 5710-5715.

[31] Tudoroiu, N., and Khorasani, K., "Satellite fault diagnosis using a bank of interacting Kalman filters," *Aerospace and Electronic Systems, IEEE Transactions on,* Vol. 43, No. 4, 2007, pp. 1334-1350.

[32] Azarnoush, H., and Khorasani, K., "Fault detection in spacecraft attitude control system," *IEEE International Conference on Systems, Man and Cybernetics, ISIC,* 2007, pp. 726-733.

[33] Chen, W., and Saif, M., "Observer-Based Fault Diagnosis of Satellite Systems Subject to Time-Varying Thruster Faults," *Journal of Dynamic Systems, Measurement, and Control,* Vol. 129, No. 3, 2007, pp. 352-356.

[34] Frank, P.M., and Koppen-Seliger, B., "Recent trends in fault diagnosis: a survey," *27th Edition International Conference in Automation,* Vol. 2, 1996, pp. 709-721.

[35] Patton, R.J., Lopez-Toribio, C.J., and Uppal, F.J., "Artificial intelligence approaches to fault diagnosis for dynamic systems," *Journal of Applied Mathematics and Computer Science,* Vol. 9, No. 3, 1999, pp. 471-518.

[36] Angeli, C., "Online expert systems for fault diagnosis in technical processes," *Expert Systems,* Vol. 25, No. 2, 2008, pp. 115-132.

[37] Li, Z.Q., Ma, L., and Khorasani, K., "A Dynamic Neural Network-based Reaction Wheel Fault Diagnosis for Satellites," *International Joint Conference on Neural Networks (IJCNN '06),* 2006, pp. 3714-3721.

[38] Sobhani-Tehrani, E., and Khorasani, K., "Fault Diagnosis of Nonlinear Systems Using a Hybrid Approach," *Lecture Notes in Control and Information Sciences,* Vol. 383, Springer, 2009, pp. 268.

[39] Venkatasubramanian, V., and Vaidyanathan, R., "Diagnosing noisy process data using neural networks," *IFAC/IMACS Symposium on Fault Detection,* No. 6, 1991, pp. 547-552.

[40] Wong, J., McDonald, K., and Palazoglu, A., "Classification of process trends based on fuzzified symbolic representation and hidden Markov models," *Journal of Process Control,* Vol. 8, No. 5, 1998, pp. 395-408.

[41] MacQueen, J.B., "Some methods for classification and analysis of multivariate observations," *5-th Berkeley Symposium on Mathematical Statistics and Probability,* Vol. 1, Berkeley; University of California Press, 1967, pp. 281-297.

[42] Vingron, M., and Argos, P., "A fast and multiple sequence alignment algorithm," *Computer Applications in the Biosciences,* Vol. 5, No. 5, 1989, pp. 115-121.

[43] Sun, W., Palazoglu, A., and Romagnoli, J.A., "Detecting abnormal process trends by wavelet-domain hidden Markov models," *AIChE Journal,* Vol. 49, No. 1, 2003, pp. 140-150.

[44] Colomer, J., Melendez, J., and Gamero, F.I., "Pattern recognition based on episodes and DTW. Application to diagnosis of a level control system," *16th International Workshop on Qualitative Reasoning,* Barcelona, Catalonia, Spain, 2002, pp. 37-43.

[45] Ke, Z., Bin, J., and Peng, S., "Application of the Wavelet transform in machine condition monitoring and fault diagnostics," *Mechanical Systems and Signal Processing,* Vol. 18, 2004, pp. 199-221.

[46] Freestone, J.W., and Jenkins, E.G., "Diagnosis of cylinder power faults in diesel engines by flywheel speed measurement," *Institution of Mechanical Engineers. Part D, Journal of Automobile Engineering,* Vol. 200, University of Technology, Loughborough, England, 1986, pp. 37-43.

[47] Azarnoush, H., and Khorasani, K., "Fault detection in spacecraft attitude control system," *IEEE International Conference on Systems, Man and Cybernetics,* 2008, pp. 726-733.

[48] Lee, J.D., Park, B.G., Kim, T.S., "Simple fault detection algorithm of BLDC motor based on operating characteristic," *IEEE Power Electronics Specialists Conference,* 2008, pp. 643-646.

[49] Gustafsson, F., "Adaptive Filtering and Change Detection," John Wiley & Sons, 2000, pp. 510.

[50] El-Mowafy, A., and Mohamed, A., "Attitude Determination from GNSS Using Adaptive Kalman Filtering," *The Journal of Navigation,* Vol. 58, 2005, pp. 135-148.

[51] Fathabadi, V., Shahbazian, M., Salahshour, K., "Comparison of Adaptive Kalman Filter Methods in State Estimation of a Nonlinear System," *World Congress on Engineering and Computer Science,* Vol. 2, 2009, pp. 884-891.

[52] Hide, C., Moore, T., and Smith, M., "Adaptive Kalman Filtering for Low-cost INS/GPS," *The Journal of Navigation,* Vol. 56, No. 1, 2003, pp. 143-152.

[53] Peng, Y., Youssouf, A., Arte, P., "A Complete Procedure for Residual Generation and Evaluation with Application to a Heat Exchanger," *IEEE Transactions on Control Systems Technology,* Vol. 5, No. 6, 1997, pp. 542-555.

[54] Min, Z.H., and Sun, L.M., "A Novel Non-Parametric Sequential Probability Ratio Test Method for Structural Condition Assessment," *SPIE - The International Society for Optical Engineering,* Vol. 7650, 2010, pp. 76502O-76502O-9.

[55] Shao, J.Y., Wang, R.X., and Xu, M.Q., "Application of Bayesian network in model-based fault diagnosis," *Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition),* Vol. 40, No. 1, 2010, pp. 234-237.

[56] Cen, Z.H., Wei, J.L., Jiang, R., "Real-time fault diagnosis of satellite attitude control system based on sliding-window wavelet and DRNN," *Chinese Control and Decision Conference,* 2010, pp. 1218-1222.

[57] Cheng, Y.H., Jiang, B., Yang, M.K., "Self-Organizing Fuzzy Neural Network-Based Actuator Fault Estimation for Satellite Attitude Systems," *Journal of Applied Sciences,* Vol. 28, No. 1, 2010, pp. 72-76.

[58] Barua, A., Sinha, P., and Khorasani, K., "A diagnostic tree approach for fault cause identification in the attitude control subsystem of satellites," *IEEE Transactions on Aerospace and Electronic Systems,* Vol. 45, No. 3, 2009, pp. 983-1002.

[59] Bialke, B., "High fidelity mathematical modeling of reaction wheel performance," *21st Annual American Astronautical Society Rocky Mountain Guidance and Control Conference,* 1998, pp. 483-496.

[60] Eberhart, R., and Kennedy, J., "*A New Optimizer Using Particle Swarm theory*," *Sixth International Symposium on Micro Machine and Human Science (MHS '95),* 1995, pp. 39-43.

[61] Alger, M., and Richard, W., "Design and Development of Power and Attitude Control Subsystems for Ryesat," *Ryerson Univesity Technical Report,* Vol. 1, No. 1, 2006,

[62] Tudoroiu, N., and Khorasani, K., "State estimation of the vinyl acetate reactor using unscented Kalman filters (UKF)," *International Conference on Industrial Electronics and Control Applications,* 2006, pp. 05E1175-05E1179.

[63] Julier, S., Uhlmann, J., and Durrant-Whyte, H.F., "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on Automatic Control,* Vol. 45, No. 3, 2000, pp. 477-482.

[64] Simandl, M., and Straka, O., "Sampling densities of particle filter: a survey and comparison," *American Control Conference,* 2007, pp. 4437-4442.

[65] Crisan, D., "A survey of convergence results on particle filtering methods for practitioners," *IEEE Transactions on Signal Processing,* Vol. 50, No. 3, 2002, pp. 736-746.

[66] Schetz, J.A., and Rogers, R.M., "Applied Mathematics in Integrated Navigation Systems," Vol. 1, AIAA (American Institute of Aeronautics & Astronautics) Inc., 2007,

[67] Trudinger, C.M., Raupach, M.R., Rayner, P.J., "Using the Kalman filter for parameter estimation in biogeochemical models," *Environmetrics 2008,* Vol. 19, 2008, pp. 849-870.

[68] Kim, K.Y., Lee, J.T., Yu, D.K., "Parameter estimation of noisy passive telemetry sensor system using unscented Kalman filter," *International Conference on Future Generation Communication and Networking,* 2007, pp. 433-438.

[69] Shellenbarger, J.C., "Estimation of covariance parameters for adaptive Kalman filter," *National Electronics Conference,* Vol. 22, 1966, pp. 77-79.

[70] Sims, F.L., Lainiotis, D.G., and Magill, D.T., "Recursive algorithm for calculation of adaptive Kalman filter weighting coefficients," *IEEE Transactions on Automatic Control,* Vol. AC-14, 1969, pp. 215-218.

[71] Kirlin, R.L., and Moghaddamjoo, A., "Conventional and robust adaptive Kalman filtering: a survey," *IEEE International Symposium on Circuits and Systems,* 1987, pp. 77.

[72] Reina, G., Vargas, A., Nagatani, K., "Adaptive Kalman Filtering for GPS-based Mobile Robot Localization," *IEEE International Workshop on Safety, Security and Rescue Robotics,* 2007, pp. 78-83.

[73] Yang, Y., and Gao, W., "An optimal adaptive Kalman filter," *Journal of Geodesy,* Vol. 80, No. 4, 2006, pp. 177-183.

[74] Zhang, S.W., Qiu, C.J., and Xu, Q., "Estimating soil water contents from soil temperature measurements by using an adaptive Kalman filter," *Journal of Applied Meteorology,* Vol. 43, No. 2, 2004, pp. 379-389.

[75] Tzou, H.K., and Lin, Y.T., "The tracking of a manoeuvring object by using an adaptive Kalman filter," *Institution of Mechanical Engineers, Part I (Journal of Systems and Control Engineering),* Vol. 215, 2001, pp. 125-130.

[76] Escamilla-Ambrosio, P.J., and Mort, N., "Multi-sensor data fusion architecture based on adaptive Kalman filters and fuzzy logic performance assessment," *Fifth International Conference on Information Fusion,* Vol. 2, 2002, pp. 1542-1549.

[77] Myers, K.A., and Tapley, B.D., "Adaptive sequential estimation with unknown noise statistics," *IEEE Transactions on Automatic Control,* Vol. AC-21, No. 4, 1976, pp. 520-523.

[78] Zhou, J., Knedlik, S., and Loffeld, O., "INS/GPS Tightly-coupled Integration using Adaptive Unscented Particle Filter," *The Journal of Navigation,* Vol. 63, 2010, pp. 491-511.

[79] Lee, T.S., "Theory and application of adaptive fading memory Kalman filters," *IEEE Transactions on Circuits and Systems,* Vol. 35, 1988, pp. 474-477.

[80] Shi, Y., and Eberhart, R.C., "*A modified particle swarm optimizer*," *IEEE International Conference on Evolutionary Computation,* 1998, pp. 69-73.

[81] Kennedy, J., "The particle swarm: social adaptation of knowledge," *IEEE International Conference on Evolutionary Computation,* Ieee, 1997, pp. 303-308.

[82] Kao, C.C., Chuang, C.W., and Fung, R.F., "The self-tuning PID control in a slider–crank mechanism system by applying particle swarm optimization approach," *Mechatronics,* Vol. 16, No. 8, 2006, pp. 513-522.

[83] Mukherjee, V., and Ghoshal, S., "Intelligent particle swarm optimized fuzzy PID controller for AVR system," *Electric Power Systems Research,* Vol. 77, No. 12, 2007, pp. 1689-1698.

[84] Pontani, M., and Conway, B., "Particle Swarm Optimization Applied to Space Trajectories," *Journal of Guidance, Control, and Dynamics,* Vol. 33, No. 5, 2010, pp. 1429-1441.

[85] Poli, R., "Analysis of the publications on the applications of particle swarm optimisation," *Journal of Artificial Evolution and Applications,* Vol. 2008, 2008, pp. 3.

[86] Clerc, M., "Particle swarm optimization," *Recherche,* Vol. 67, 2006, pp. 02.

[87] Premalatha, K., and Natarajan, A.M., "Hybrid PSO and GA for Global Maximization," *Int. J. Open Problems Compt. Math,* Vol. 2, No. 4, 2009, pp. 597-608.

[88] Potter, M., and De Jong, K., "A cooperative coevolutionary approach to function optimization," *Parallel Problem Solving from Nature (PPSN III),* Vol. 866, 1994, pp. 249-257.

[89] Zielinski, K., and Laur, R., "Stopping criteria for a constrained single-objective particle swarm optimization algorithm," *Informatica (Ljubljana),* Vol. 31, No. 1, 2007, pp. 51-59.