ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

NANOSATELLITE ATTITUDE ESTIMATION VIA TRIAD-AIDED KALMAN FILTERS

M.Sc. THESIS

Mehmet Asım GÖKÇAY

Department of Aeronautical and Astronautical Engineering

Aeronautical and Astronautical Engineering Programme

JUNE 2020

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY

NANOSATELLITE ATTITUDE ESTIMATION VIA TRIAD-AIDED KALMAN FILTERS

M.Sc. THESIS

Mehmet Asım GÖKÇAY
(511171116)

Department of Aeronautical and Astronautical Engineering

Aeronautical and Astronautical Engineering Programme

Thesis Advisor: Prof. Dr. Cengiz HACIZADE

JUNE 2020

ISTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

TRIAD METODU DESTEKLİ KALMAN SÜZGEÇLERİ İLE NANO UYDULARDA YÖNELİM KESTİRİMİ

YÜKSEK LİSANS TEZİ

Mehmet Asım GÖKÇAY
(51171116)

Uçak ve Uzay Mühendisliği Anabilim Dalı

Uçak ve Uzay Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Cengiz HACIZADE

HAZİRAN 2020

Mehmet Asım Gökçay, a M.Sc. student of ITU Graduate School of Science Engineering and Technology student ID 511171116, successfully defended the thesis entitled "NANOSATELLITE ATTITUDE ESTIMATION VIA TRIAD-AIDED KALMAN FILTERS", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Cengiz HACIZADE** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Fikret ÇALIŞKAN** Istanbul Technical University	.....
	Dr. Öğr. Üyesi Janset DAŞDEMİR Yıldız Technical University	.....

Date of Submission : 11 June 2020
Date of Defense : 26 June 2020

To my family,

FOREWORD

First of all, I would like to express my gratefulness to my thesis advisor Prof. Dr. Cengiz HACIZADE for his guidance and support during the development of this thesis.

I am also thankful to my friends Egemen Aydın, Berke Ölçücüoğlu and Sinan Arda Üner for their valuable friendship and Turkish Aeropace for their support. But most importantly, I would like to express my gratitude to my family for their great support and encouragements. I would not be able to be at where I am without their love, efforts and guidance.

June 2020 Mehmet Asım Gökçay

TABLE OF CONTENTS

Page

ABBREVIATIONS

A **:** Attitude Matrix
AEKF : Adaptive Extended Kalman Filter
AFKF **:** Adaptive Fading Kalman Filter
AU **:** Astronomical Unit
AUKF : Adaptive Unscented Kalman Filter
DCM **:** Direction Cosine Matrix
ECEF **:** Earth Centered Earth Fixed
ECI **:** Earth Centered Inertial
EKF **:** Extended Kalman Filter
JD **:** Julian Date
LEO **:** Low Earth Orbit
MEKF **:** Multiplicative Extended Kalman Filter
NASA **:** National Aeronautics and Space Administration
PF **:** Particle Filter
RAAN **:** Right Ascension of The Ascending Node
RMSE **:** Root Mean Square Error
SO(3) **:** The Special Orthogonal Group for Three-Dimensional Space
SPD **:** Semi Positive Definite
SVD **:** Singular Value Decomposition
TRIAD **:** Tri-axial Attitude Determination
UKF **:** Unscented Kalman Filter
VSOP **:** Variations Séculaires des Orbites Planétaires
w.r.t **:** with respect to

SYMBOLS

b	**:** Bias
B	**:** Magnetic Field
$\tilde{\mathbf{C}}_k$	: Estimated Innovation Covariance
$\mathbf{C}_k$	: Innovation Covariance
F	**:** Jacobian
h	**:** Angular Momentum
H	**:** Measurement Jacobian
I	**:** Identity Matrix
J	**:** Moment of Inertia
K	**:** Kalman Gain
L	**:** Dimension of the state vector
$\mathbf{M}_e$	**:** Magnetic Dipole Moment
P	**:** Covariance Matrix
$\tilde{\mathbf{P}}_k$	:Estimated Covariance
Q	**:** Process Noise
q	**:** Quaternion
R	**:** Measurement Noise
S	**:** Sun Direction Vector
T	**:** External Torque
t	**:** Time
w	**:** Angular Velocity w.r.t. inertial frame
W	**:** Weight
x	**:** State
$\mathbf{Y}_i$	**:** Propagated Sigma Vector
z	**:** Measurement
Z	**:** Propagated Measurement Sigma Vector
χ^2	: Chi Square
$\boldsymbol{\Psi}$	: Yaw Angle
$\boldsymbol{\alpha}_k$	: Scaling Factor
$\boldsymbol{\delta}$	: Kronecker Symbol

ζ_k : Forgeting Factor

η : Noise

η_s : Level of Significance

θ : Pitch Angle

ι : Inclination

λ : Scaling Parameter

$\lambda_{ecliptic}$: Ecliptic Longitude of Sun

$\lambda_{M_{sun}}$: Mean Longitude of Sun

μ : Earth Gravitational Constant

ν : Innovation

σ : Standart Deviation

χ_i : Sigma Points

ω : Angular Velocity w.r.t. Body Frame

ϵ : Magnetic Dipole Tilt

ϵ_e : Obliquity of Ecliptic

ϕ : Roll Angle

LIST OF TABLES

Page

LIST OF FIGURES

Page

NANOSATELLITE ATTITUDE ESTIMATON VIA TRIAD-AIDED KALMAN FILTERS

SUMMARY

Increasing demand for the space operations, space industry turns its face to cost effective solutions. Small satellites, due to their size and cost, are receiving interest from many organizations. In 2018, NASA sent two MarCO cubesats to Mars. Their mission was to relay the landing vehicle data back to Earth. Restricted size comes with its own challenges. The amount of attitude determination and control equipment that can be placed in small satellites are considerably lower than a the regular size satellite. In this work, using common sensors, couple of filters are design to overcome to attitute determination problem.

Two of most common sensors that are being used in nanosatellites are magnetometers and sun sensors. Magnetic dipole model is selected for magnetic field model. VSOP87 theory is used for sun direction vector. Using these two models, sensor measurement models have been established. For attitude representation of the spacecraft euler angles are selected. Using these angles, equations of motion of the spacecraft are obtained.

One of the earliest attitude determination method is algebraic method. Using sun sensor as the first triad, additional two triads have been constructed. Constructed three vectors form a direction cosine matrix. Body angles are obtained from this matrix. In order to increase the accuracy of the satellite motion parameters, three different methods have been analyzed.

Extended Kalman filter, unscented Kalman filter and adaptive fading Kalman filters are derived and designed for the system. Comparison of these three filters are studied. Designing the filters, a new sensor type, gyroscopes are used. Measurement model of the gyroscope is obtained.

Body angles that are produced by algebraic method are used as linear measurements to the Kalman filters. Hence, two methods are integrated for achieving better accuracy for body angles and angular velocities. Analytic jacobian matrices for EKF have been calculated. Magnetometer and gyroscope biases are estimated. Instead of using magnetometer measurements, magnetometer is corrected with estimated biases. Same process has been applied to the gyroscope measurements. Sensor measurements can be corrupted for many reasons. Aadaptive filters are applied to both EKF and UKF to increases robustness of the both filters. All of the algorithms are designed in MATLAB. Simulation are also conducted at this package program.

TRIAD METODU DESTEKLİ KALMAN SÜZGEÇLERİ İLE NANO UYDULARDA YÖNELİM KESTİRİMİ

ÖZET

İlk çağlardan beri uzay insanoğlu için hep bir gizem kaynağı olmuştur. Eski Çin imparatorluklarının barut kullanarak basit roketler yaptıkları bilinsede asıl ciddi çalışmalar 19. yüzyılın sonları doğru gerçekleşmiştir. 2. Dünya Savaşı'ndaki V2 roketlerının başarısı çoğu ülkenin bu tip cihazlara ilgisini arttırmıştır. Soğuk Savaş'ın başlamasıyla uzay artık askeri bir ortama dönüştü. Bu yıllarda uzaya atılan Sovyet yapımı uydu, Sputnik ile birlikte uzay çalışmaları ciddi ivme kazandı. ABD'nin Mercury ve Gemini görevleri ve Apollo programı ile bu ivme en yüksek noktaya ulaşmıştır. Dr. Kalman'ın lineer olmayan fonksiyonların kestrimi için hazırladığı Kalman süzgeci bu yıllarda ortaya çıkmıştır. İlk çıkışıyla birlikte süzgeç, özellikle havacılık ve uzay sektöründe kendine sağlam bir yer edinmiştir. Yıllar geçtikçe uzay araçları gelişen teknolojiyle küçülmüş ve ucuzlamıştır. Küpsatların ortaya çıkmasıyla birlikte artık üniversitelerin bile kendi içlerinde uydu yapabileceği bir seviyeye gelinmiştir.

Bu çalışmada kütleleri 1 ile 10 kilogram arası değişen nano uyduların yönelimine yönelik süzgeç tasarımları yapılmıştır. İlk önce uydu dinamik, kinematik denklemleri ve dönüşüm matrisi türetilmiştir. Daha sonra nano uydularda sıklıkça kullanılan sensörler verilmiştir. Bunlar; güneş sensörü, manyetometre ve jiroskoptur. Tezde ilk olarak Güneş'in yönünün belirlenmesi için bir algoritma verilmiştir. Bu algoritma sayesinde Dünya ile Güneş arasındaki yön vektörü hesaplanarak sensör ölçme modeli oluşturuluyor. Burada uydu ile Dünya arasında mesafe ihmal ediliyor.Güneş yörünge modeli için VSOP87 kullanılmıştır. Manyetometre ölçüm modeli için manyetik alan dipol modeli kullanılmuştır. Jiroskop modeli için uydu matematik modelinden yararlanılmıştır.

En eski yönelim belirleme tekniklerinden olan TRIAD metoduyla Güneş sensörü ve manyetometre ölçümlerinden faydalanarak bir yönelim matrisi oluşturulmuştur. TRIAD metodu Güneş sensörü ölçüm vektörünü sabit tutarak manyetometre ölçüm vektörü ile üç adet vektör oluşturarak ortogonal bir yönelim matrisi oluşturur. Bu matris yardımıyla sapma, yunuslama ve yuvarlanma açıları elde edilmiştir. TRIAD algoritması ilk yönelim belirleme için düzgün sonuçlar vermiştir. Bu algoritmadan elde edilen açılar daha sonra Kalman süzgeçlerine lineer olarak dahil edilmiştir. Süzgeç tasarım bölümünde üç adet süzgeç üzerinde durulmuştur.Bunlar; genişletilmiş Kalman süzgeci, sezgisiz Kalman süzgeci ve bu süzgeçlerin adaptif tipleri. İlk olarak genişletilmiş Kalman filtresi türetilip simülasyon için gerekli olan kovaryans ve gürültü matrisleri belirlenmiştir. İki farklı simülasyon programı oluşturulmuştur. İlk programda sadece açılar ve açısal hızlar kestirilmiştir. TRIAD metodundan gelen açılar, jiroskoptan gelen açısal hızlar ve manyetometreden gelen manyetik alan ölçümleri ölçüm matrisini oluşturmuştur. Ardından simülasyon sonuçları ve hata grafikleri ayrı bir bölümde belirtilmiştir. İkinci programda ise sensör önyargıları da

durum matrisine dahil edilmiştir. Böyle on iki elemanlı bir durum matrisi oluşturulmuştur. Kovaryans ve gürültü matrisleri oluşturulduktan sonra yine aynı şekilde grafikler verilmiştir.

Genişletilmiş Kalman süzgecine nazaran sezgisiz Kalman süzgeci lineer olmayan fonksiyonları daha iyi kestirebilmektedir. Bu süzgeçte de iki program yapılmıştır. İlk programda on üç, ikinci programda ise yirmi beş sigma noktası kullanılmıştır. Ölçüm matrisleri genişletilmiş Kalman süzgeci programlarında olduğu gibidir. İki programın grafikler ve hata değerleri ayrı bölümde verilmiştir. Kullanılan başlangıç hızlarına bağlı olarak iki süzgecin de performansları farklılık göstermiştir. Genel olarak sezgisiz Kalman süzgecinin genişletilmişe nazaran daha iyi sonuçlar verdiği gözlemlenmiştir. Genişletilmiş Kalman süzgecinin türetilmesindeki kısmi türevlerin alınması hataya çok açıktır. Bu yüzden sezgisiz Kalman süzgeci tasarım açısından kolay sağlasa da daha fazla işlem gücüne ihtiyaç duymaktadır. Günümüzde gelişen ve küçülen işlemciler yüksek boyutlu süzgeçlerin uydu üzerinde yapılabilmesini sağlamaktadır.

Süzgeç tasarımının son ayağında ise adaptif süzgeçler çalışılmıştır. Adaptif süzgeçlerin genel olarak iki kullanım alanı mevcuttur. Bunlar; hatalı sensör ölçüm değerlerinin belirlenip göz ardı edilmesi ve araç parametrelerinin tahminidir. Bu çalışmada hatalı sensör verilerinin düzeltilmesi üzerine çalışılmıştır. Hem genişletilmiş hem de sezgisiz Kalman süzgeçlerine eklemeler yapılarak adaptif hale getirilmişlerdir. Sensör hatalarını iki kategoriye ayrılmıştır, gürültü ekleme yoluyla oluşan hatalar ve yanlış önyargının sebep olduğu hatalar. İki durumda değerlendirilmeye alınmıştır. Ayrıca hatanın ekleme yöntemleri de incelenmiştir. Hata temelde iki şekilde eklenmiştir. İlk olarak anlık gürültü veya önyargı eklemesi yapılmıştır. Bunun için dört farklı yapı kurulmuştur. Rastgele değerler seçilerek matematik modele gürültü ve önyargı eklemeleri yapılmıştır. İlk olarak manyetometre verisine gürültü eklemesi yapılmış ve genişletilmiş Kalman süzgecinin nasıl kestirme yapıldığına bakılmıştır. Daha sonra süzgeç adaptif hale getirilerek düzeltme oranları incelenmiştir. Ayrı işlem jirokop için de yapılmıştır. Bu işlem sezgisiz Kalman süzgeci için aynı şekilde gerçekleşmiştir. Son olarak, sensör ölçme değerlerine önyargı eklenmiştir. Aynı şekilde orijinal süzgeçlerin nasıl tepki verdiklerine baktıktan sonra süzgeçler adaptif hale getirilip düzeltme oranları incelenmiştir. İkinci hata ekleme yöntemi olarak sürekli hata test edilmiştir. Bunun için ölçümde belirli bir aralık seçilip, bu aralığa gürültü veya önyargı eklemesi yapılmıştır. Adaptif filtreler ile oluşturulan grafikler ayrı bir bölümde gösterilmiştir. Sonuç olarak adaptif filtreler belirli bir miktardaki hatayı önemli ölçüde telafi edebileceği sonucuna ulaşılmıştır fakat hata sayıları arttıkça süzgeçlerin kestirim performansları da düşmüştür.

Lineer olmayan durumların kompleks bir hal almaya başlaması sezgisiz Kalman filtreyi genişleştilmiş filtrenin önüne koymaktadır. TRIAD algoritmasını düzeltmede sezgisiz filtre daha iyi bir performans ortaya koymaktadır. Ölçüm sayısının kestirimlere olan etkisini ölçmek için durum vektörleri aynı bırakılarak manyetometre ölçümleri de altı elemanlı durum vektörleriyle kestirime sokulmuştur. Bu işlem hem genişletilmiş hem de sezgisiz filtreler ile yapılmıştır. Sonuçlar ortalama karesel hata değerlerine bakılarak karşılaştırılmıştır. Tezin son bölümünde adaptif yöntemlerin simülasyon sonuçlarına yer verilmiştir. Hata modellemesinin ilk durumunda hata anlık olarak rastgele seçilen noktalara uygulanmış ve adaptif yöntemler ile bu hata düzeltilmeye çalışılmıştır. İkinci durumda ise hata zamana yayılmıştır. Her iki durum için de bozulmuş ve düzeltilmiş durumlar grafiklerde karşılaştırılmış olarak verilmiştir. Adaptif yöntemde algoritmanın bozulmayı anlayabilmesi için bir istatiksel yöntem kullanılmıştır. Buna göre bir istatistik fonksiyonu kullanılarak bir değer elde

edilmiştir. Ki-kare tablosu kullanılarak durum vektörüne bağlı olarak bir değer seçilmiştir. Bu değer ile istatistik fonksiyonundan gelen değerler karşılaştırılmış ve ölçümün hatalı olup olmadığı incelenmiştir. Eğer fonksiyondan gelen değer büyük ise ölçüm hatalı kabul edilip adaptif yöntem devreye sokulmuştur. Adaptif yöntemin ortaya koyduğu sonuca bakıldığı zaman anlık hatalar yüksek doğrulukla düzeltilebiliyorken devamlı hataların düzeltilmesi bozuntunun ilk bölümlerinde kabul edilebilir seviyede iken son bölümlerde kötüleşmektedir.

1. INTRODUCTION

1.1 Purpose of Thesis

The aims of the thesis are given as follows;

- To establish sensor models.
- To find attitude matrix using TRIAD algorithm.
- To integrate TRIAD algorithm with different version of Kalman filter.
- To compare Extended and Unscented Kalman filter for a nonlinear problem.
- To use an adaptive filter to provide robustness for corrupted measurements.

1.2 Literature Review

Kalman filter has been the backbone of the space industry since the publication of the Dr. Kalman's work (Kalman, 1960). Many different versions of the Kalman filter have been derived. It is important even more in small satellite applications to find suitable filter since sensor variety is less than full size spacecrafts. Star tracker and other high-accurate sensors often can not be used due their cost and size.

Harold Black published an algorithm called TRIAD algorithm in 1964 (Black, 1964). It is the earliest algorithm that was published to find satellite attitude with two measurements.In 1965, Grace Wahba suggested her famous problem (Wahba, 1965). Solution methods such as q-method (Wertz, 1978) and Quest algorithm developed by Schuster have been widely used (Schuster and Oh, 1981). A computationally expensive method, SVD, is also published (Landis, 1988). Many of the metioned methdods are coupled with Kalman filters for higher accuracy (Cilden-Guler, Soken and Hajiyev, 2018), (Cilden-Guler and Hajiyev, 2017). In this work algebraic method is used with sun sensor and magnetometer measurements.

EKF is probably the most used version of the Kalman filter (Schmidt, 1981). One of the earliest work of Kalman filter for attitude determination used Euler angle rotations

(Farrell, 1970). It is known that all three-parameter representations of the special orthogonal group suffer from singularity and discontuinity problems. To overcome this challenge new representation methods have been studied (Shuster, 1993). Quaternions have become the most used form of the attitude representations. Euler angles, Rodrigues and modified Rodrigues parameters are avoided for most of the agile missions due to their singularity problems (Stuelpnagel, 1964). One of the many advantages of the quaternions is kinematic equations become linear and non-singular. Another advantage is that successive rotations can be represented by quaternion multiplication (Kuipers, 1999). Many different types of sensor usage on various missions proved that a unit vector representation is more convinient (Sedlak and Chu, 1993).

Multiplicative EKF is offered as a solution to SO(3) constaints of the regular EKF. Attitude is represented as a product of estimated attitude and a deviation from that estimate (Crassidis, Markley and Cheng, 2007). The normalized quaternion estimate provides nonsingular attitude representation. Main advantages of the MEKF are that covariance matrix has minimum dimensionality and as aforementioned case, dealing with only very small angles provides global nonsingularity (Markley, 2003). Second order versions of MEKF are also available (Vathsal, 1987). Performance problems of EKF, mainly due to linearization phase, has led researches to derivations of EKF. Backward-Smoothing EKf combines the some parts of the EKF with smoothing algorithms. It relinearize past finite member of measurements when a new measurement is needed (Psiaki, 2005). EKF-like nonlinear estimators based on H_∞ control design technique have been suggested. Their main advantage is that noises within the algorithms do not have to be white noise (Markley, Berman and Shaked, 1994). Also, in the last two decades new approaches have been suggested for replacing EKF. Unscented Kalman filter is one of them (Julier, Uhlmann and Durrant-White, 1995). UKF uses the unscented transformation to achieve high-order approximations of the nonlinear functions in order to estimate mean and covariance of the state vector. Filter uses predefined number of sigma points to approximate Gaussian distribution. Each of the sigma points are propagated through the propagation functions (Wan and van der Merwe, 2001). The most computationally costly part of the UKF is calculating sigma points in each time step. Cholesky factorization is efficient for matrix square root operations. In square root UKF, instead of updating the covariance matrix directly,

square of the covariance matrix is propagated. Square root operation is conducted only at the initialization step. Using Cholesky factor updating, square root matrix is propagated (van der Merwe and Wan, 2001). Sigma points only represents the Gaussian distribution. But mean and covariance alone can not represents general probability distribution. One of the Monte Carlo based nonlinear estimators, particle filter may be used for solving the probability distribution problem. PFs generate random samples using pseudorandom number generators to estimate distributions (Doucet, de Freitas and Gordan, 2001). In highly nonlinear cases, PFs are superior to other nonlinear filters. Main issues for particle filter are their high computational requirements and complexity. In one of the suggested PFs, attitude is only propagated through motion equations while measurements are only used for updating the weight (Cheng and Crassidis, 2004). A genetic form of PF wass suggested for estimating the measurement noise and gyro biases (Oshman and Carmi, 2005).

Adaptive approaches are mainly divided into two parts (Soken and Hacizade, 2019). In the first part, adaptive filtering used to tune process and measurement noise covariances (Ma and Ng, 2002), (Hajiyev, Soken and Cilden-Guler, 2019). A version of the adaptive EKF is suggested for colored noises (Lam and Wu, 1998). Another commonly use case for adaptive filters is fault tolerance (Mehra et. al, 1995). Small spacecrafts are more prone to fail due to their interconnected design. Adaptive filters are used to detect faulty measurements (Rapoport and Oshman, 2002). The second part of the adaptive approaches is to estimate system parameters. Using flight data, system dynamic model's fidelity can be enhanced by suggested filter (Psiaki, 2003).

1.3 Thesis Outline

Chapter 2 gives the coorinate systems that are used in this thesis. It explains different attitude representation methods. It derives the attitude matrix and satellite's equations of motion. In chapter 3, three different sensor models are derived. Magnetic field and sun direction models are explained. Main part of the thesis is chapter 4.The algebraic method, EKF and UKF are derived. Also adaptive versions of EKF and UKF are explained. In each subsections, simulation results for each method are presented. And for the final chapter, chapter 5 gives conclusion.

2. SATELLITE MATHEMATICAL MODEL

2.1 Coordinate Systems

In this thesis, three main coordinate systems are used. These are satellite body frame, orbit reference frame and earth inertial frame. Definitions for these systems are given below.

Earth Centered Inertial Frame: The origin of this frame is located at the center of the Earth. x – axis points toward the Vernal equinox while z – axis points toward geographic North Pole. The y – axis completes the axis triad. (Figure 2.1)

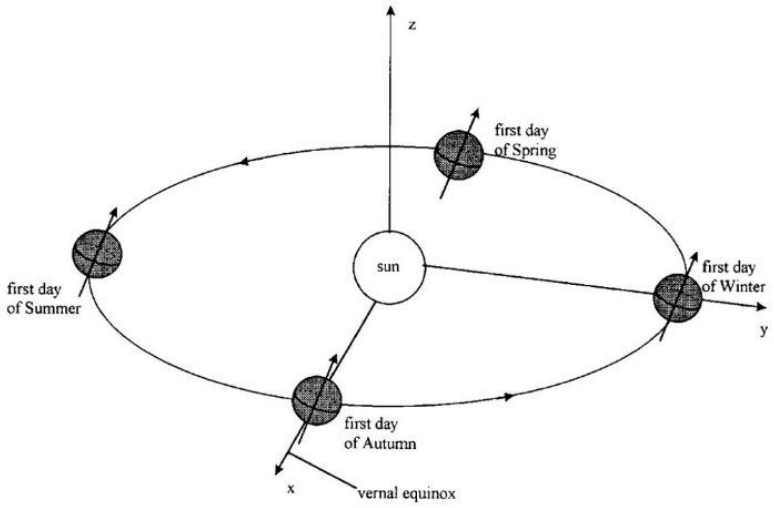

Figure 2.1 : ECI frame (Bao and Tsui, 2005).

Orbit Reference Frame*:* The origin of this frame is located in satellite's center of mass. z – axis is in nadir direction, points toward Earth. x – axis is in the same direction of velocity vector of the satellite in circular orbit. y – axis is in the direction of negative orbital normal vector. (Figure 2.2)

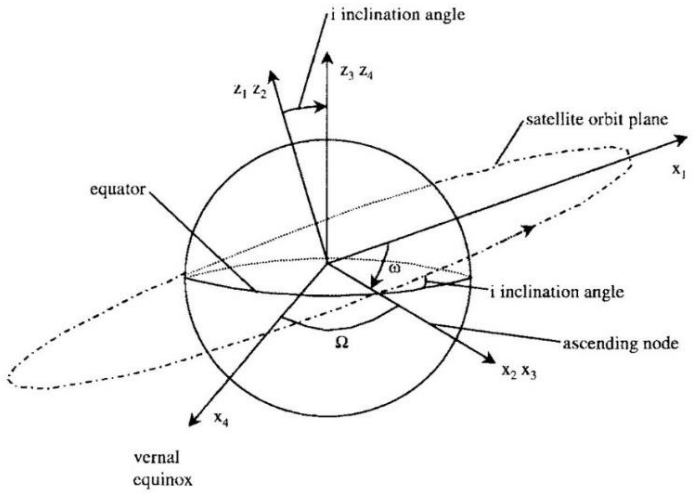

Figure 2.2 : Reference frame (Bao and Tsui, 2005).

Satellite body frame: The origin of this frame is located in center of mass of the satellite. The axes match with satellite's inertial axes. This frame is related to orbit reference frame by direction cosine matrix which is constructed by Euler angles. If the direction cosine matrix is unity matrix, this frame coincides with orbital reference frame. (Figure 2.3)

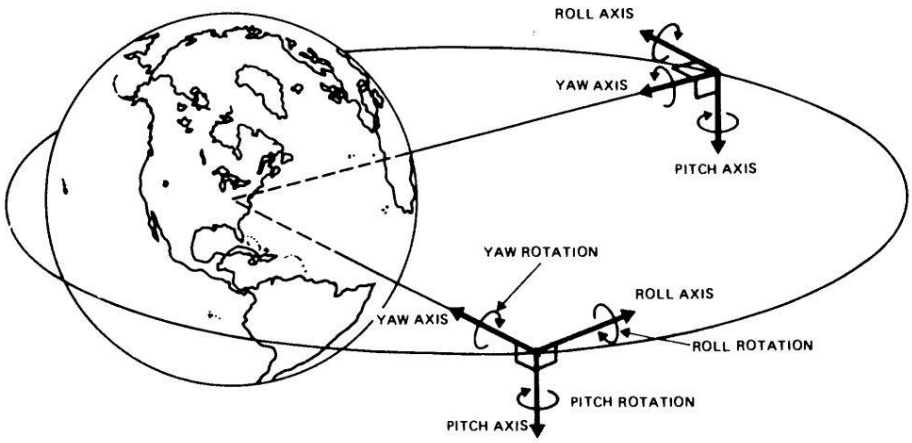

Figure 2.3 : Body frame (Wertz, 1978).

2.2 Attitude Representation

Two of the most common attitude representations, Euler Angles and quaternions are given in this section.

2.2.1 Quaternions

Quaternion representation serves as an alternative to Euler angle rotation (Hamilton, 1866). Following four parameters form the quaternion,

$$q = q_4 + iq_3 + jq_2 + kq_3 \tag{2.1}$$

where i,j,k represents hyperimaginary numbers. q_4 is the scalar part of the quaternion and $iq_3+jq_2+kq_3$ is the imaginary part. If the vector q corresponds to vector part of q, then the alternative way of representing the quaternion would be (Wertz, 1978),

$$\mathbf{q} = q(q_4, \mathbf{q}) \tag{2.2}$$

Four paramaters can be defined by following equations

$$q_1 = e_1 \sin\left(\frac{\Phi}{2}\right) \tag{2.3}$$

$$q_2 = e_2 \sin\left(\frac{\Phi}{2}\right) \tag{2.4}$$

$$q_3 = e_3 \sin\left(\frac{\Phi}{2}\right) \tag{2.5}$$

$$q_4 = \cos\left(\frac{\Phi}{2}\right) \tag{2.6}$$

where e is rotation axes and Φ is rotation angle about respected axis. It is important that quaternions are not independent. They satisfy the condition that is given below

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1 \tag{2.7}$$

2.2.2 Euler angles

Euler angles are a set of three angles. Let angles be denoted by a,b,c. These angle describe a three consecutive rotations from a reference frame.

I. Initial rotation from friex reference frame about about any x,y,z axes by a. New coordinate system can be denoted as x’,y’,z’

II. Rotation about any x′,y′,z′ axes by b. New system can be denoted by x″,y″,z″

III. Final rotation about any x″,y″,z″ axes by z. Last coordinate system can be denoted as u,v,w.

In this thesis, final coordinate system coincides with satellite body frame.

2.3 Rotation

Let û,v̂ and ŵ be a orthogonal triad system. They satisfy,

$$\hat{u} + \hat{v} = \hat{w} \tag{2.8}$$

The basic problem of satellite attitude determination is to define the satellite's orientation using orthogonal triad system. If the component of this triad is known, then the orientation will be fixed. This requires defining a "attitude matrix", A,

$$A = \begin{bmatrix} u_1 & v_1 & w_1 \\ u_2 & v_2 & w_2 \\ u_3 & v_3 & w_3 \end{bmatrix} \tag{2.9}$$

where each column specifies the components of the triad along with three axes. Each elements of the A matrix represents cosine angle of the angle between body and reference frames. For this reason, A matrix is also referred as direction cosine matrix. In the essence, dcm maps reference frame to body frame. If p is a vector in reference frame, then

$$Ap = \begin{bmatrix} u_1 & v_1 & w_1 \\ u_2 & v_2 & w_2 \\ u_3 & v_3 & w_3 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} \hat{u}.p \\ \hat{v}.p \\ \hat{w}.p \end{bmatrix} = \begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix} \tag{2.10}$$

Right side of the equation shows components of the vector p along the u,v and w triad. In his famous theorem, Euler states any rotation can be describe by three rotation angles (Euler, 1775). Below, rotation matrices about all three axes by θ are given.

Rotation about axis 1 by θ

$$R_1(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \tag{2.11}$$

- Rotation about axis 2 by θ

$$R_2(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.12}$$

- Rotation about axis 3 by θ

$$R_3(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

In this thesis, the DCM is constructed using 2-1-3 rotation.

$$\begin{aligned} A_{213}(\theta,\phi,\psi) &= A_3(\psi)A_1(\phi)A_3(\theta) \\ &= \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \end{aligned} \tag{2.14}$$

After matrix multiplication, dcm has been derived.

$$\begin{bmatrix} c\psi c\theta + s\phi s\psi s\theta & c\phi s\psi & c\theta s\phi s\psi - c\psi s\theta \\ c\psi s\phi s\theta - c\theta s\psi & c\phi c\psi & s\psi s\theta + c\psi c\theta s\phi \\ c\phi s\theta & -s\phi & c\phi c\theta \end{bmatrix} \tag{2.15}$$

where c(.) and s(.) are cosine and sine functions respectively. From 2-1-3 rotation matrix, Euler angles can be extracted from matrix elements with equations below.

$$\psi = \operatorname{atan}2\left(\frac{A(1,2)}{A(2,2)}\right) \tag{2.16}$$

$$\theta = \operatorname{atan}\left(\frac{A(3,1)}{A(3,3)}\right) \tag{2.17}$$

$$\phi = \operatorname{atan}\left(\frac{-A(3,2)\cos(\psi)}{A(2,2)}\right) \tag{2.18}$$

where atan and atan2 are both arctangent functions. atan2 is the arc tangent functionwith two arguments for complete covarage of four quadrants.

2.4 Equations of Motion

2.4.1 Kinematic equations

Obtaining a frame from another frame is possible with rotation matrices. Using the fact that angular velocities are additive, angle rates can be determined in terms of these velocities. Considering 2-1-3 rotation, initially frame rotates about y axis by θ. Second rotation is about x′ by ϕ and final rotation is about w by ψ. Summing angular velocity vectors for each rotation,

$$\omega = \dot{\theta}\vec{y} + \dot{\phi}\vec{x}' + \dot{\psi}\vec{w} \tag{2.19}$$

Taking the components of ω in $\vec{u}, \vec{v}, \vec{w}$

$$\begin{aligned} \omega_u &= \dot{\theta}\vec{y}\vec{u} + \dot{\phi}\vec{x}'\vec{u} \\ \omega_v &= \dot{\theta}\vec{y}\vec{v} + \dot{\phi}\vec{x}'\vec{v} \\ \omega_w &= \dot{\theta}\vec{y}\vec{w} + \dot{\phi}\vec{x}'\vec{w} + \dot{\psi} \end{aligned} \tag{2.20}$$

In order to derive kinematic equations, (2.20) needs to be determined. $\vec{y}\vec{u}$, $\vec{y}\vec{v}$ and $\vec{y}\vec{w}$ can be calculated from DCM. $\vec{y}$ indicates that second column of the DCM.

$$\vec{y}\vec{u} = \cos\phi\sin\psi \tag{2.21}$$

$$\vec{y}\vec{v} = \cos\phi\cos\psi \tag{2.22}$$

$$\vec{y}\vec{w} = -\sin\phi \tag{2.23}$$

For second rotation, $R_3(\psi)R_1(\phi)$ matrix needs to be calculated. Taking first column of the second rotation matrix

$$\vec{x}'\vec{u} = \cos\psi \tag{2.24}$$

$$\vec{x}'\vec{v} = -\sin\psi \tag{2.25}$$

$$\vec{x}'\vec{w} = 0 \tag{2.26}$$

Hence, angular velocity equations become

$$\omega_u = \dot{\theta}\cos\phi\sin\psi + \dot{\phi}\cos\psi \tag{2.27}$$

$$\omega_v = \dot{\theta}\cos\phi\cos\psi - \dot{\phi}\sin\psi \tag{2.28}$$

$$\omega_w = -\dot{\theta}\sin\phi + \dot{\psi} \tag{2.29}$$

In matrix form,

$$\omega_{bo} = \begin{bmatrix} \cos\phi\sin\psi & \cos\psi & 0 \\ \cos\phi\cos\psi & -\sin\psi & 0 \\ -\sin\phi & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} \tag{2.30}$$

Taking inverse of the matrix, Euler angle rates can be determined in terms of angular velocities in body frame with respect to reference frame (Wertz 1978). Rate equations are given below

$$\dot{\phi} = \omega_u \cos\psi - \omega_v \sin\psi \tag{2.31}$$

$$\dot{\theta} = (\omega_u \sin\psi + \omega_v \cos\psi)\sec\phi \tag{2.32}$$

$$\dot{\psi} = \tan\phi(\omega_u \sin\psi + \omega_v \cos\psi) + \omega_w \tag{2.33}$$

Rotations need to be defined with respect to inertial frame. For this reason, a transformation matrix is needed. Using the relation below, angular velocities with respect to inertial frame can be calculated.

$$\begin{bmatrix} \omega_u \\ \omega_v \\ \omega_w \end{bmatrix} = \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} - A \begin{bmatrix} 0 \\ -\omega_0 \\ 0 \end{bmatrix} \tag{2.34}$$

where $\omega_0 = \sqrt{\frac{\mu}{r^3}}$ is angular velocity at the altitude r. μ is gravitational constant of the Earth and r is the distance between center of mass of the satellite and the Earth, $r = |\vec{r}|$

2.4.2 Satellite dynamics

Attitude dynamics is related with time derivative of the angular momentum vectror.If the origin of the body frame is selected as the center of mass, c, then,

$$h_c = J\vec{w} \tag{2.35}$$

where h is angular momentum vector, w is angular velocity vector in body frame wrt inertial frame and J is moment of inertia matrix. J is defined as,

$$J = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \tag{2.36}$$

The relation between time derivative of the angular momentum and the angular velocity is,

$$\dot{\vec{h}}_c = \dot{\vec{h}}_b + \vec{w} \times (\vec{h}_c) \tag{2.37}$$

where $\dot{\vec{h}}_b$ is time derivative of angular momentum as seen in body-fixed frame. Knowing the time rate of change of the angular momentum is equal to external torque, T_c and using (2.35)

$$T_c = \dot{\vec{h}}_b + \vec{w} \times (J\vec{w}) \tag{2.38}$$

Rewriting this equation considering time derivative of the w is same in both body-fixed and reference frames,

$$\dot{\vec{h}}_b = J\dot{\vec{w}} \tag{2.39}$$

$$\dot{\vec{w}} = J^{-1}\left[T_c - \vec{w} \times (J\vec{w})\right] \tag{2.40}$$

The external torque can be defined as sum of the gravity gradient torque, aerodynamic torque, magnetic torque and solar pressure disturbance. In this thesis, only magnetic torque is considered as external torque.

$$T_c = T_m \tag{2.41}$$

Finally, rewriting (2.38) for discrete time,

$$w_{x_{k+1}} = w_{x_k} + \frac{\Delta t}{J_x}\left[w_z w_y (J_y - J_z) + T_m\right] \tag{2.42}$$

$$w_{y_{k+1}} = w_{y_k} + \frac{\Delta t}{J_y}\left[w_x w_z (J_z - J_x) + T_m\right] \tag{2.43}$$

$$w_{z_{k+1}} = w_{z_k} + \frac{\Delta t}{J_z}\left[w_x w_y (J_x - J_y) + T_m\right] \tag{2.44}$$

Equations (2.27)-(2.29) and (2.42)-(2.44) describe the satellite attitude motion.

3. MODELS OF SENSOR MEASUREMENTS

3.1 Magnetometer Measurement Model

Magnetometer is the most commonly used sensor particularly in nanosatellite applications. For magnetic field vector, dipole model is used. Sensor model is given below,

$$\begin{bmatrix} B_x(\phi,\theta,\psi,t) \\ B_y(\phi,\theta,\psi,t) \\ B_z(\phi,\theta,\psi,t) \end{bmatrix} = A \begin{bmatrix} B_1(t) \\ B_2(t) \\ B_3(t) \end{bmatrix} + b_m + \eta_m \tag{3.1}$$

where $B_1(t)$, $B_2(t)$ and $B_3(t)$ indicates Earth magnetic field vector components in orbit frame and given by (Wertz 1978),

$$\begin{aligned} B_1(t) = \frac{M_e}{r^3}\Big\{ & \cos(\omega_0 t)\left[\cos(\varepsilon)\sin(\iota) - \sin(\varepsilon)\cos(\iota)\cos(\omega_e t)\right] \\ & -\sin(\omega_0 t)\sin(\varepsilon)\sin(\omega_e t)\Big\} \end{aligned} \tag{3.2}$$

$$B_2(t) = -\frac{M_e}{r^3}\left[\cos(\varepsilon)\cos(\iota) + \sin(\varepsilon)\sin(\iota)\cos(\omega_e t)\right] \tag{3.3}$$

$$\begin{aligned} B_3(t) = \frac{2M_e}{r^3}\Big\{ & \sin(\omega_0 t)\left[\cos(\varepsilon)\sin(\iota) - \sin(\varepsilon)\cos(\iota)\cos(\omega_e t)\right] \\ & +2\cos(\omega_0 t)\sin(\varepsilon)\sin(\omega_e t)\Big\} \end{aligned} \tag{3.4}$$

where $M_e = 7.943\times10^{15}$ *Wb.m* is magnetic dipole moment of the Earth, $\varepsilon = 11.7^\circ$ is the magnetic dipole tilt, $\omega_e = 7.29\times10^{-5}$ *rad/s* is the spin rate of the Earth, ι is the orbit inclination and $\mu = 3.98601\times10^{14}\ m^3/s^2$ is the Earth Gravitational constant.

$B_x(\phi,\theta,\psi,t)$, $B_y(\phi,\theta,\psi,t)$ and $B_z(\phi,\theta,\psi,t)$ show the Earth magnetic field vector components in body frame as a function of body angles and time. The magnetometer bias vector is given as $b_m = \begin{bmatrix} b_x & b_y & b_z \end{bmatrix}^T$. Bias vector is model as rate of change of the bias vector is constant in time.

$$\dot{b}_m = 0 \tag{3.5}$$

The noise term, η_m, is added linearly to the model and modeled as zero mean Gaussian white noise with the characteristic of

$$E\left[\eta_{1k}\eta_{1j}^T\right] = I_{3x3}\sigma_m^2\delta_{kj} \tag{3.6}$$

where I_{3x3} is the identity matrix, σ_m is the standart deviation of magnetometer errors and δ_{kj} is the Kronecker symbol.

3.2 Sun Sensor Measurement Model

In order to construct a sun sensor model, sun direction vector is used. Using VSOP87 theory, a direction cosine matrix is calculated which shows the sun's position relative to Earth in ECI frame (Bretagnon and Francou, 1988).

$$\begin{bmatrix} s_{B_x} \\ s_{B_y} \\ s_{B_z} \end{bmatrix} = A \begin{bmatrix} s_{E_1} \\ s_{E_2} \\ s_{E_3} \end{bmatrix} + \eta_s \tag{3.7}$$

Construction of sun direction vector, s_E requires two assumptions.Comparing the distance between Sun-Earth, 1 AU, and Earth-satellite, satellite altitude is negligible. Therefore satellite's sun direction vector is always parallel to Earth's sun direction vector. The other assumption is taking the right ascension node of Sun's orbit as zero.

Model is using Julian Date (JD). The reference epoch of the first order model is the January 1st 2000, 12:00:00 pm. Converting this date to JD would be 2451545. Satellite's epoch is selected as March 16th 2017, 22:46:22. Frist step of calculating the direction vector is to find mean anomaly of the Sun. All of the angles that are given below is in degrees.

$$T_{TDB} = \frac{JD_{UTC} - 2451545}{36525} \tag{3.8}$$

$$M_{Sun} = 357.5277233° + 35999.05034T_{TDB} \tag{3.9}$$

Secondly, ecliptic longitude of the Sun, $\lambda_{ecliptic}$ is calculated.

$$\lambda_{ecliptic} = \lambda_{M_{Sun}} + 1.914666471\sin\left(M_{Sun}\right) + 0.019994643\sin\left(2M_{Sun}\right) \tag{3.10}$$

where $\lambda_{M_{Sun}}$ is the mean longitude of the sun. It can be calculated with the equation below.

$$\lambda_{M_{Sun}} = 280.460 + 36000.770T_{sat} \tag{3.11}$$

Initially, T_{sat} is satellite's epoch. It will increases by one second until satellite reach at the end of its first period. Lastly, the angle between Earth's orbit and equator planes, obliquity of the ecliptic needs to be calculated.

$$\varepsilon = 23.439291 - 0.0130042T_{TDB} \tag{3.12}$$

The sun direction vector,

$$s_E = \begin{bmatrix} \cos\lambda_{ecliptic} \\ \sin\lambda_{ecliptic}\cos\varepsilon \\ \sin\lambda_{ecliptic}\sin\varepsilon \end{bmatrix} \tag{3.13}$$

The noise term, η_s, is added linearly to the model and modeled as zero mean Gaussian white noise with the characteristic of

$$E\left[\eta_{1k}\eta_{1j}^T\right] = I_{3x3}\sigma_s^2\delta_{kj} \tag{3.14}$$

where I_{3x3} is the identity matrix, σ_s is the standart deviation of sun sensor errors and δ_{kj} is the Kronecker symbol.

3.3 Gyro Measurement Model

The gyro model is constructed with satellite dynamic equations. A commonly used model for gyro measurements given by

$$\omega_{BI} = \omega_{BI} + b_g + \eta_g \tag{3.15}$$

where b_g is the gyro bias vector and the η_g modeled as zero mean Gaussian white noise with the characteristic of

$$E\left[\eta_{1k}\eta_{1j}^T\right] = I_{3x3}\sigma_g^2\delta_{kj} \tag{3.16}$$

where I_{3x3} is the identity matrix, σ_g is the standart deviation of gyro errors and δ_{kj} is the Kronecker symbol. Bias vector is model as rate of change of the bias vector is constant in time.

$$\dot{b}_g = 0 \tag{3.17}$$

4. FILTER DESIGN

In this chapter, the design methods that are used for attitude determination process are given. Initially, a deterministic attitude determination process, the algrebraic method, also known as TRIAD method, is given. The outputs of the TRIAD method are then used in Kalman filter variants, extended and unscented Kalman filters.

After each subsection, results are presented. For simulation, an imaginary satellite is selected. Its orbital attributes are given in Table 4.1.

Table 4.1 : Orbital Parameters.

RAAN (deg)	Argument of Perigee (deg)	Inclination (deg)	Altitude (km)
310.94	207.4	97.65	400

Using (2.42)-(2.44) and (2.34), satellite's mathematical model is constructed. Satellite's inertia matrix is given below.

$$J = \begin{bmatrix} 2.1\times10^{-3} & 0 & 0 \\ 0 & 2\times10^{-3} & 0 \\ 0 & 0 & 1.9\times10^{-3} \end{bmatrix} \tag{4.1}$$

4.1 The Algebraic Method

A rotation matrix describes the attitude of a spacecraft with respect to a known reference frame. It takes at least two measured vectors to determine the orientation of the vehicle. This direction cosine matrix has nine elements but only three quantites are sufficient to built the matrix. Therefore, with two measurements, four different quantites are obtained. This leads the problem to be overdetermined.The TRIAD algorithm discards the part of the measurements so that a solution can be found. (Black, 1964)

The algebraic method constucts two triads of orthonormal vectors. In this thesis, two triads are expressed by sun sensor and magnetometer unit vectors in body and

reference frames. Let magnetometer measurement vector is denoted by B and sun sensor vector is denoted by S. For initial base vector, more accurate sensor is selected as to be exact.

$$\vec{u} = \vec{S} \tag{4.2}$$

$$\vec{u}_b = \vec{S}_b \tag{4.3}$$

$$\vec{u}_r = \vec{S}_r \tag{4.4}$$

b and r subscripts denote the body and reference frames. For second base vector, a unit vector that is perpendicular to first base vector is constructed.

$$\vec{v} = \vec{S} \times \vec{B} \tag{4.5}$$

$$\vec{v}_b = \frac{\vec{S}_b \times \vec{B}_b}{\left|\vec{S}_b \times \vec{B}_b\right|} \tag{4.6}$$

$$\vec{v}_r = \frac{\vec{S}_r \times \vec{B}_r}{\left|\vec{S}_r \times \vec{B}_r\right|} \tag{4.7}$$

And the final base vector to complete the triad,

$$\vec{w} = \vec{u} \times \vec{v} \tag{4.8}$$

$$\vec{w}_b = \vec{u}_b \times \vec{v}_b \tag{4.9}$$

$$\vec{w}_r = \vec{u}_r \times \vec{v}_r \tag{4.10}$$

Three base vectors form a complete orthogonal coordinate system. It is important to note that two vectors, S and B can not be parallel, $\left|\vec{S}.\vec{B}\right| < 1$.Constructing the direction cosine matrix,

$$A^{br} = \begin{bmatrix} \vec{u}_b & \vec{v}_b & \vec{w}_b \end{bmatrix} \begin{bmatrix} \vec{u}_r & \vec{v}_r & \vec{w}_r \end{bmatrix}^T \tag{4.11}$$

Equation (4.11) results in a 3x3 matrix. Using equations (2.16)-(2.18), body angles can be obtained. Final step of the algebraic method is to find covariance of the algorithm. Calculating with the formula given below, covariance matrix is established (Markley and Crassidis, 2014).

$$P = \frac{\sigma_m^2 \vec{S}_b \vec{S}_b^T + \sigma_s^2 \vec{B}_b \vec{B}_b^T}{\left\| \vec{S} \times \vec{B} \right\|} + \sigma_s^2 \vec{v}_b \vec{v}_b^T \tag{4.12}$$

Covariance and body angles that are obtained from triad algorithm are used as measurement inputs to the kalman filter. These results are combined with gyro measurements in order to achieve better accuracy.

4.1.1 Simulation

Algebraic method is one othe earliest and simplest solutions to the attitude determination problem. In order to form triads, it is crucial to select most accurate sensor as first vector. In this work, it is selected as sun direction vector. In Figure 4.1, error in attitude estimation via triad algorithm is given.

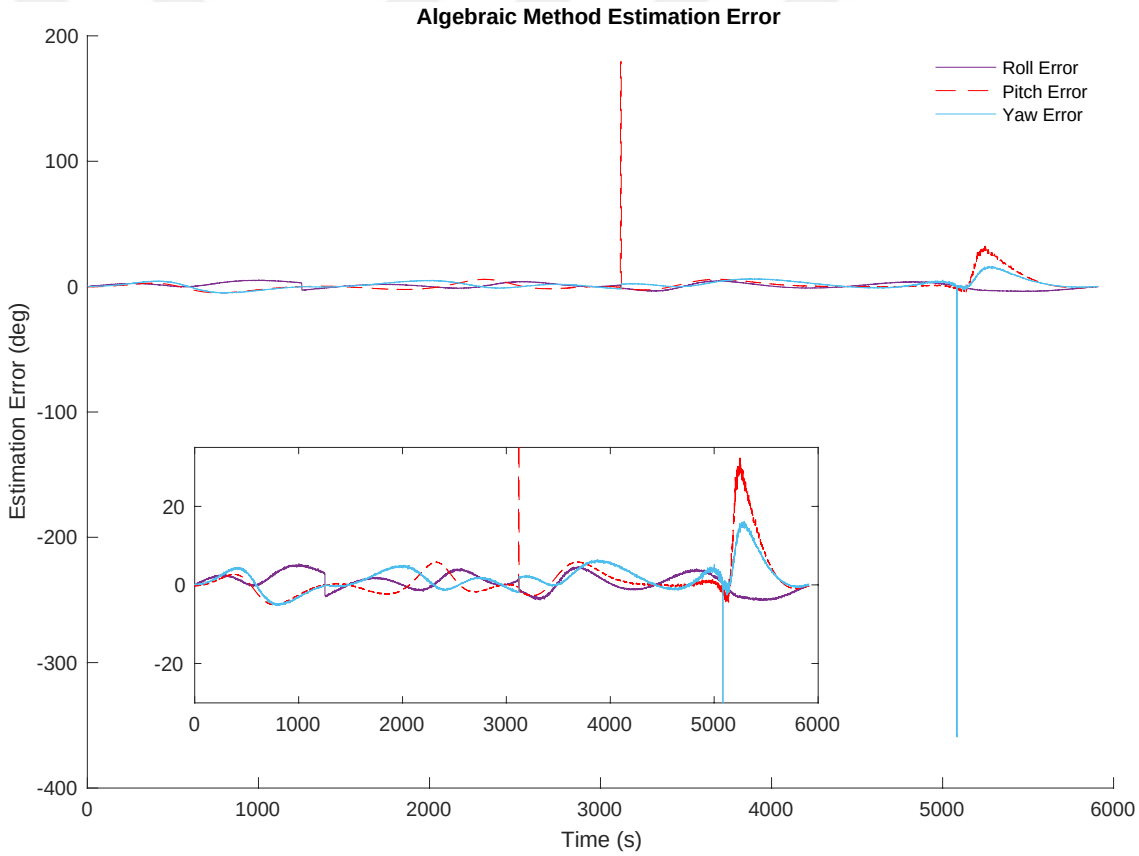

Figure 4.1 : Algebraic Method Estimation Error.

Most of the estimation errors are between -5 and +5 degrees. There are two spikes in the graph. Since the body angles are within -180/+180 degrees, algebraic method estimation's error in predicting the model can result in reaching the 180 degrees threshold early or late than mathematical model. This causes a 180 degrees gap between model and estimation. Therefore, two spikes are in acceptable region. On the

other hand, in the last part of the graph, there is a noticable error peak. In Figure 4.2, models and estimations are presented.

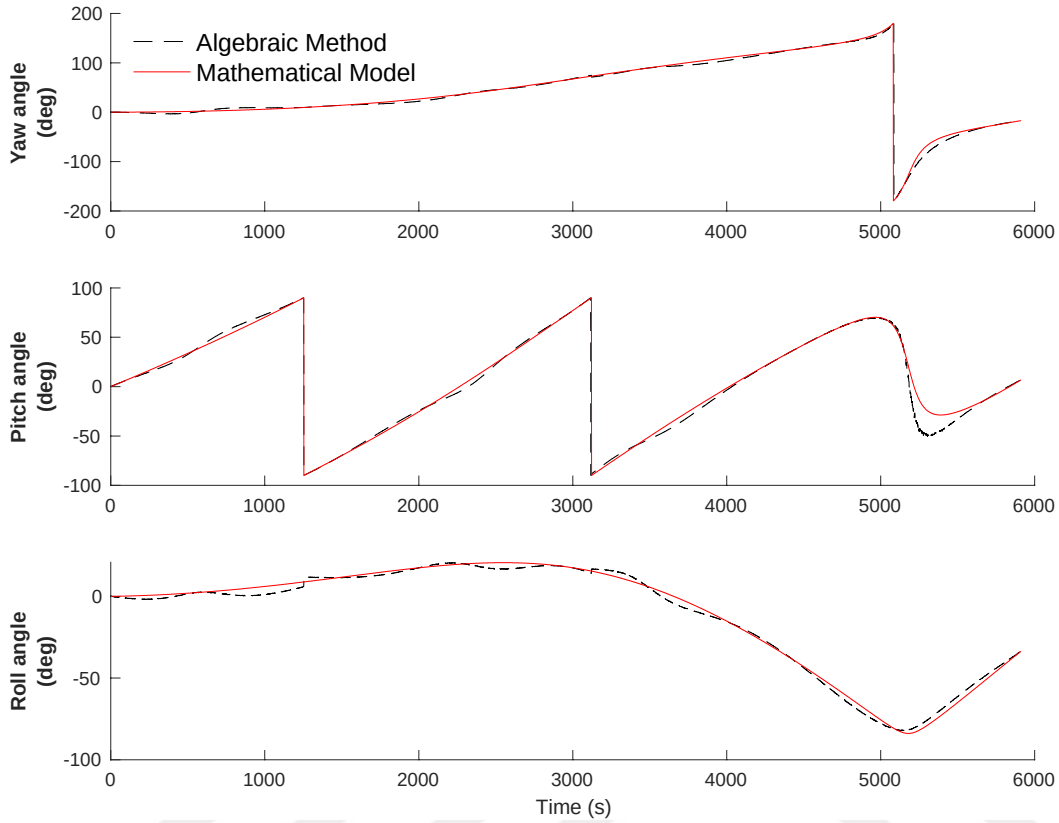

Figure 4.2 : Algebraic Method Estimation.

Figure shows that algebraic method lags behind the mathematical model. Hence, this problem creates the error pattern seen in Figure 4.1. Considering its simplicity and computational effectiveness, this method is valuable for initial attitude guesses. RMSE analysis of the algebraic method is given in Table 4.2

Table 4.2 : Algebraic Method RMSE analysis.

Algebraic Method		
ϕ	θ	ψ
6.2395	7.2175	11.4611

4.2 The Extended Kalman Filter

In 1960, Dr. Kalman published his famous paper "A new Approach to Linear Filtering and Prediction". It represented a sequantial solution to the time-varying filtering problem. The filter is particularly good for dynamical systems and also removed the non-dynamical requirements of Weiner filter. (Kalman, 1960).

The extended kalman filter uses the first order Taylor series linearization for non-linear systems. This process requires two assumptions. Propagation and measurement fucntions must be differentiable. The propagation function maps the state matrix from t-1th time to tth time.

$$\hat{x}_k^- = f\left(\hat{x}_{k-1}^+\right) \tag{4.13}$$

Where superscript "-" indicates before correction step, and "+" indicates after the correction step. EKF uses covariance matrix to model the uncertainty In order to propagate the covariance matrix, a Jacobain matrix is needed. Jacobain matrix ,F, can be constructed via taking partial derivatives of propagation functions with respect to states.Writing the state equation,

$$x_k = f\left(\hat{x}_{k-1}^+ + \Delta x_{k-1}\right) + q_{k-1} \tag{4.14}$$

x_k is true state. All of the state estimations are modeled as deviations from true state.Using Taylor expansion for propagation function,

$$x_k \approx f\left(\hat{x}_{k-1}^+\right) + F\Delta x_{k-1} + q_{k-1} \tag{4.15}$$

Using (4.13),

$$\Delta x_k = x_k - \hat{x}_k^- \approx F\Delta x_{k-1} + q_{k-1} \tag{4.16}$$

Covariance matrix can be modeled as,

$$P_k^- = E\left[\Delta x_k \Delta x_k^T\right] \tag{4.17}$$

Expanding (4.17) with (4.16),

$$P_k^- = E\left[F\Delta x_{k-1}\Delta x_{k-1}^T + F\Delta x_{k-1}q_{k-1}^T + q_{k-1}\Delta x_{k-1}^T F^T + q_{k-1}q_{k-1}^T\right] \tag{4.18}$$

Expectation function's distributive property results in four separate functions. In this study, state of the spacecraft and noise is uncorrelated. Therefore,

$$E\left[\Delta x_{k-1}q_{k-1}^T\right] = 0 \tag{4.19}$$

$$E\left[q_{k-1}\Delta x_{k-1}^T\right] = 0 \tag{4.20}$$

One of the assumptions that is initially made for applying EKF is noises are gaussian white noise. So, they have zero mean.

$$E\left[q_{k-1}q_{k-1}^T\right] = Q \tag{4.21}$$

Therefore, covariance matrix equation becomes

$$\begin{aligned} P_k^- &= FE\left[\Delta x_{k-1}\Delta x_{k-1}^T\right]F^T + Q \\ &= FP_{k-1}^+F^T + Q \end{aligned} \tag{4.22}$$

(4.13) and (4.22) propagate the state and covariance to t^{th} time. EKF's strong part is to decide which is to trust, measurement or model. Propagation of the measuremet is similar to model propagation.

$$z_k = h\left(x_k\right) + r_k \tag{4.23}$$

Applying same operations to (4.23) in order to get measurement covariance matrix.

$$P_{yy} = HP_k^-H^T + R \tag{4.24}$$

where R is the measurement noise. In correction step, there is one more covariance matrix to construct kalman gain matrix, state-innovation covariance matrix.

$$\begin{aligned} P_{xy} &= E\left[\Delta x_k \Delta z_k^T\right] \\ &= E\left[\Delta x_k \left(H\Delta x_k + r_k\right)^T\right] \\ &= E\left[\Delta x_k \Delta x_k^T H^T\right] + E\left[\Delta x_k r_k^T\right] \end{aligned} \tag{4.25}$$

Knowing state and the measurement noise is uncorrelated, (4.25) becomes

$$\begin{aligned} P_{xy} &= E\left[\Delta x_k \Delta x_k^T\right]H^T \\ &= P_k^-H^T \end{aligned} \tag{4.26}$$

Kalman gain represents the trust that is given to the measurement. The uncertainty of measurement error is shown by P_{yy}. The uncertainty with state and measurement error is shown by P_{xy}.

$$K = P_{xy}P_{yy}^{-1} \tag{4.27}$$

The corrected state can be now calculated.

$$\hat{x}_k^+ = \hat{x}_k^- + K\left(z_k - h\left(x_k^-\right)\right) \tag{4.28}$$

Next step is to correct covariance matrix. Perturbation from true state can also be corrected via Kalman gain.

$$x_k = \hat{x}_k^+ + \Delta x_k^T \tag{4.29}$$

$$\Delta x_k^+ = x_k - \hat{x}_k^+ \tag{4.30}$$

Corrected state is known. Hence

$$\Delta x_k^T = x_k - \left(\hat{x}_k^- + K\Delta z_k\right) \tag{4.31}$$

$$\Delta x_k^+ = \Delta x_k^- - K\Delta z_k \tag{4.32}$$

Corrected covariance formula is similar to uncorrected one. Therefore,

$$\begin{aligned} P_k^+ &= E\left[\Delta x_k^+ \Delta x_k^{+T}\right] \\ &= E\left[\left(\Delta x_k^- - K\Delta z_k\right)\left(\Delta x_k^- - K\Delta z_k\right)^T\right] \end{aligned} \tag{4.33}$$

Rewriting the (4.33)

$$P_k^+ = P_k^- - KP_{xy} - P_{xy}K^T + \left(P_{xy}P_{yy}^{-1}\right)P_{yy}K^T \tag{4.34}$$

Last two terms cancel out. Hence,

$$\begin{aligned} P_k^+ &= P_k^- - KP_{xy}^T \\ &= P_k^- - KHP_k^- \\ &= \left(I - KH\right)P_k^- \end{aligned} \tag{4.35}$$

(4.35) is the corrected covariance.

4.2.1 Simulation

For the first case of EKF simulation, sensor biases are ignored. A 6-state model is constructed.

$$x = \begin{bmatrix} \phi & \theta & \psi & \omega_x & \omega_y & \omega_z \end{bmatrix}^T \tag{4.36}$$

There are three sensor measurement equipments for this simulation. Therefore there are 9 elements in the measurement matrix. Sun sensor and magnetometer measurement are combined with algebraic method. Body angles that are measured from algebraic method are considered as linear measurements. Gyros measure angular velocity also directly. But for the magnetometer case, magnetic field measurements are nonlinear.

While constructing the measurement jacobians, partial derivatives of (3.1) are calculated with respect to all elements of the state.

$$z = \begin{bmatrix} \phi & \theta & \psi & \omega_x & \omega_y & \omega_z & B_x & B_y & B_z \end{bmatrix}^T \tag{4.37}$$

The simulations are realized for 5910 seconds with sampling time $\Delta t = 1$. Initial values for body angles are relatively irrelevant than angular velocities. It has been seen from trials that fast initial angular velocities can cause instabilities in filter for specific Q matrices.Therefore, initial velocities are selected as,

$$\omega = \begin{bmatrix} 2.5E-5 \\ 3.3E-7 \\ 3.5E-6 \end{bmatrix} \tag{4.38}$$

Process noise covariance matrix is constructed as,

$$Q = \begin{bmatrix} 1E-3I_{3x3} & 0_{3x3} \\ 0_{3x3} & 1E-6I_{3x3} \end{bmatrix} \tag{4.39}$$

The initial value of the covariance matrix is given as $P_o = \mathrm{diag}\left(\begin{bmatrix} 10^{-1} & 10^{-1} & 10^{-1} & 10^{-3} & 10^{-3} & 10^{-3} \end{bmatrix}\right)$ where diag(.) refers to diagonal matrix. Figure 4.3-4.5 show EKF estimation of body angles comparing to mathematical model.

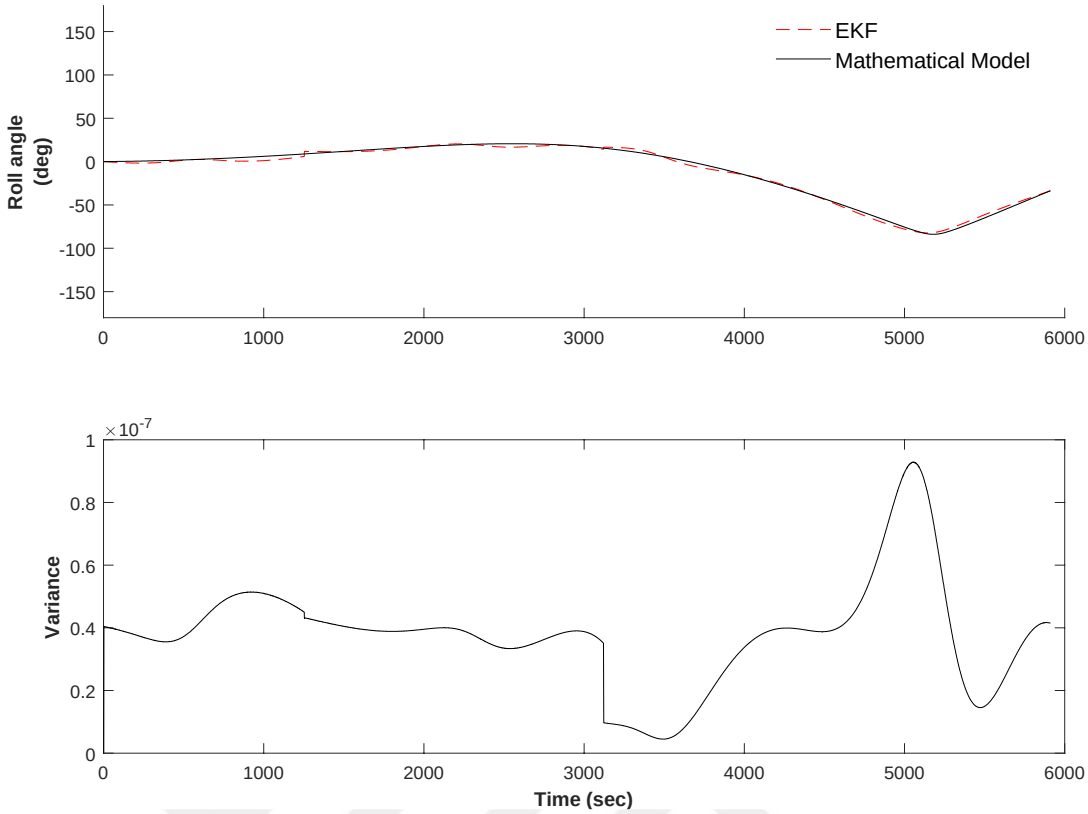

Figure 4.3 : Roll estimation with EKF.

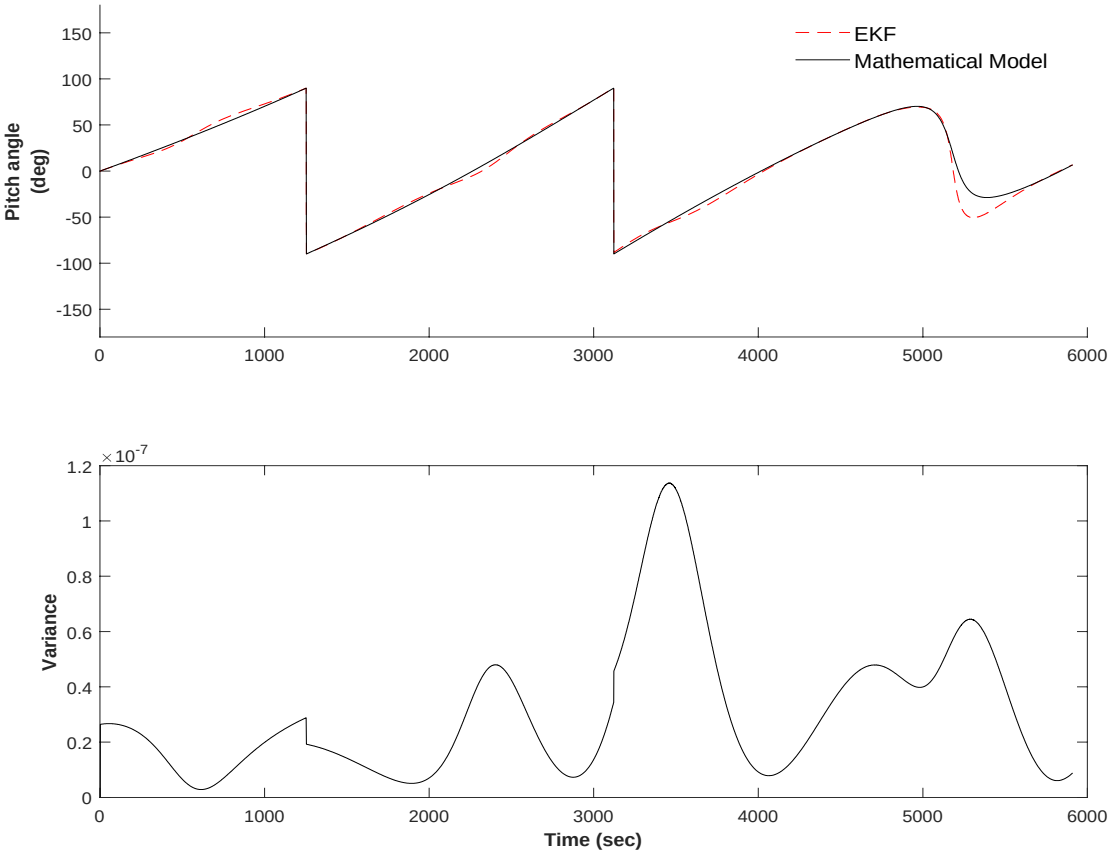

Figure 4.4 : Pitch estimation with EKF.

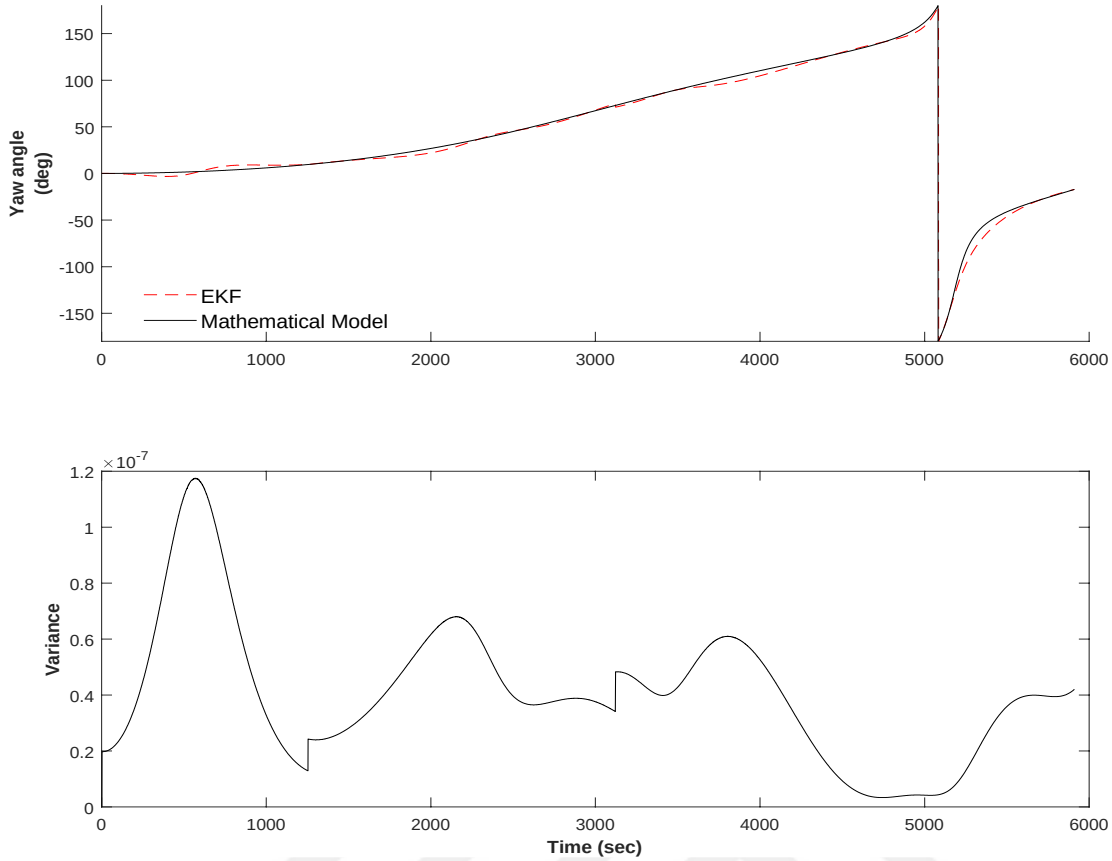

Figure 4.5 : Yaw estimation with EKF.

As seen from figures above, EKF corrected the algebraic method estimation on various parts. Graphs do not contain any major deviation apart from the last parts of the pitch angle. High yaw and pitch values are observed. These two high angles are the major source of the fluctuations that can be seen in graphs below. Figures 4.6-4.8 show the angular velocity graphs.

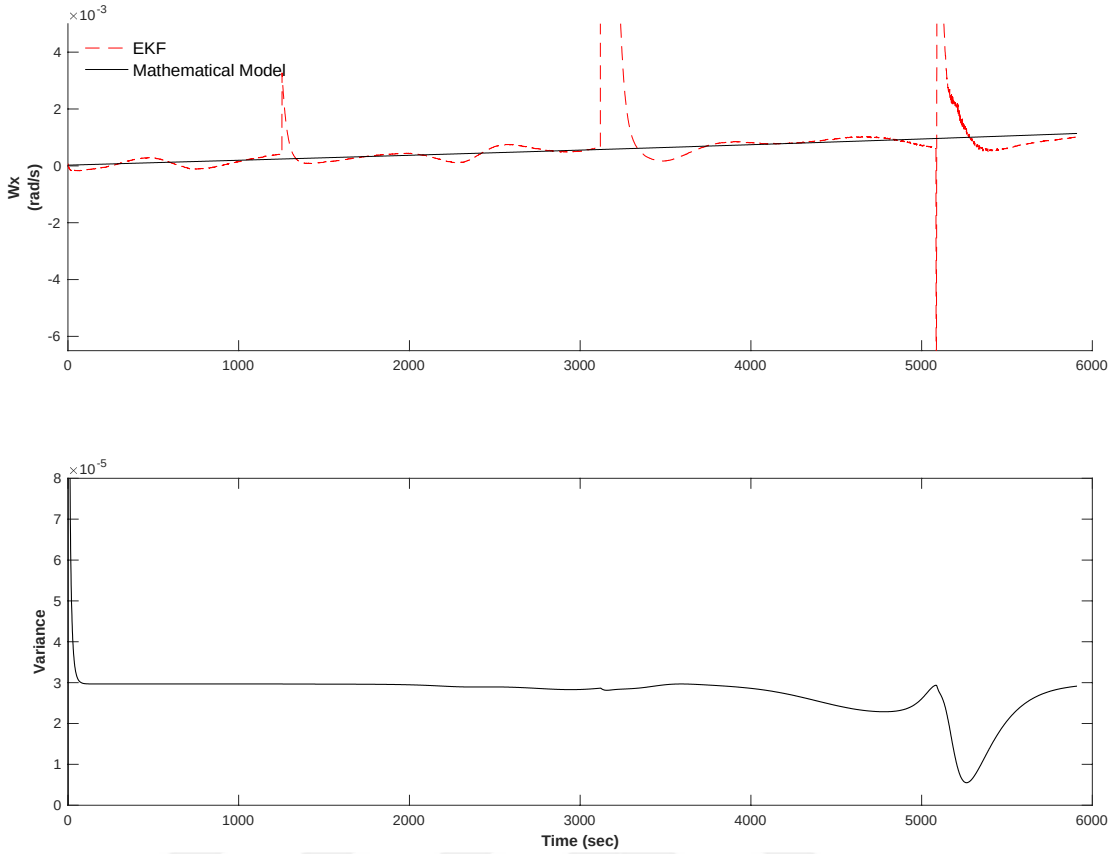

Figure 4.6 : w_x estimation with EKF.

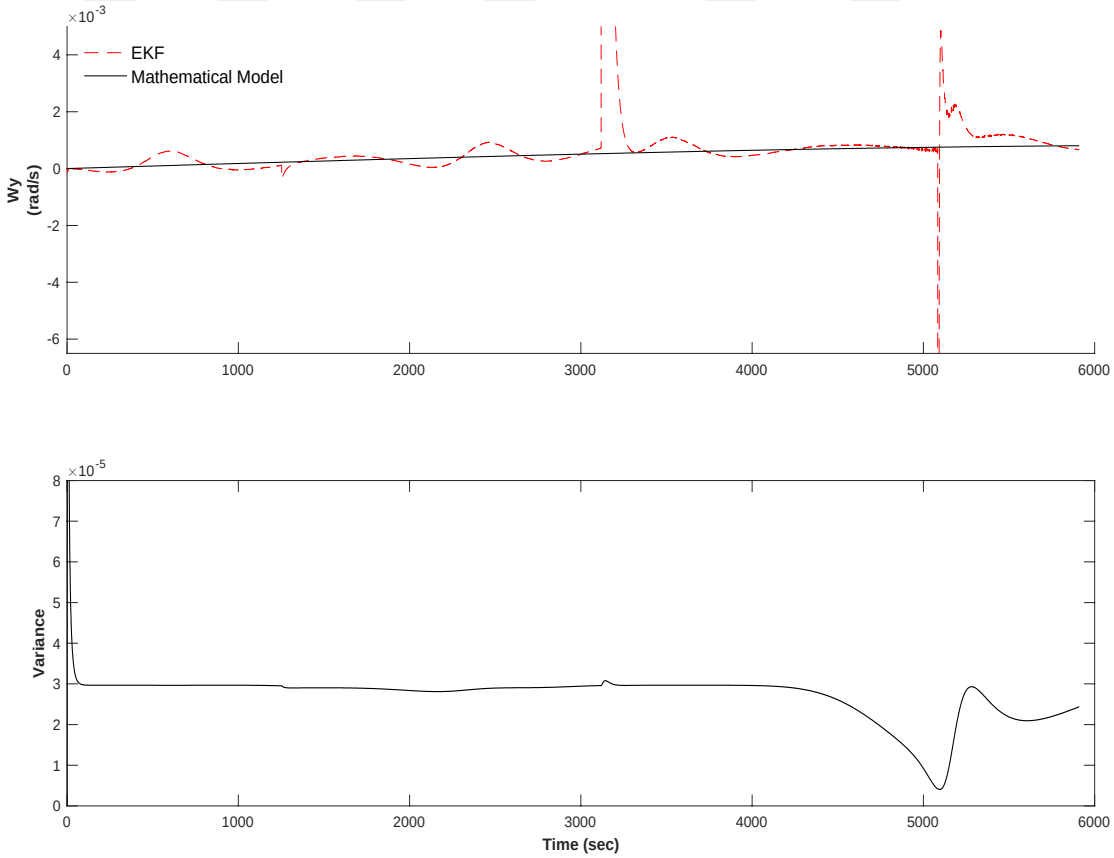

Figure 4.7 : w_y estimation with EKF.

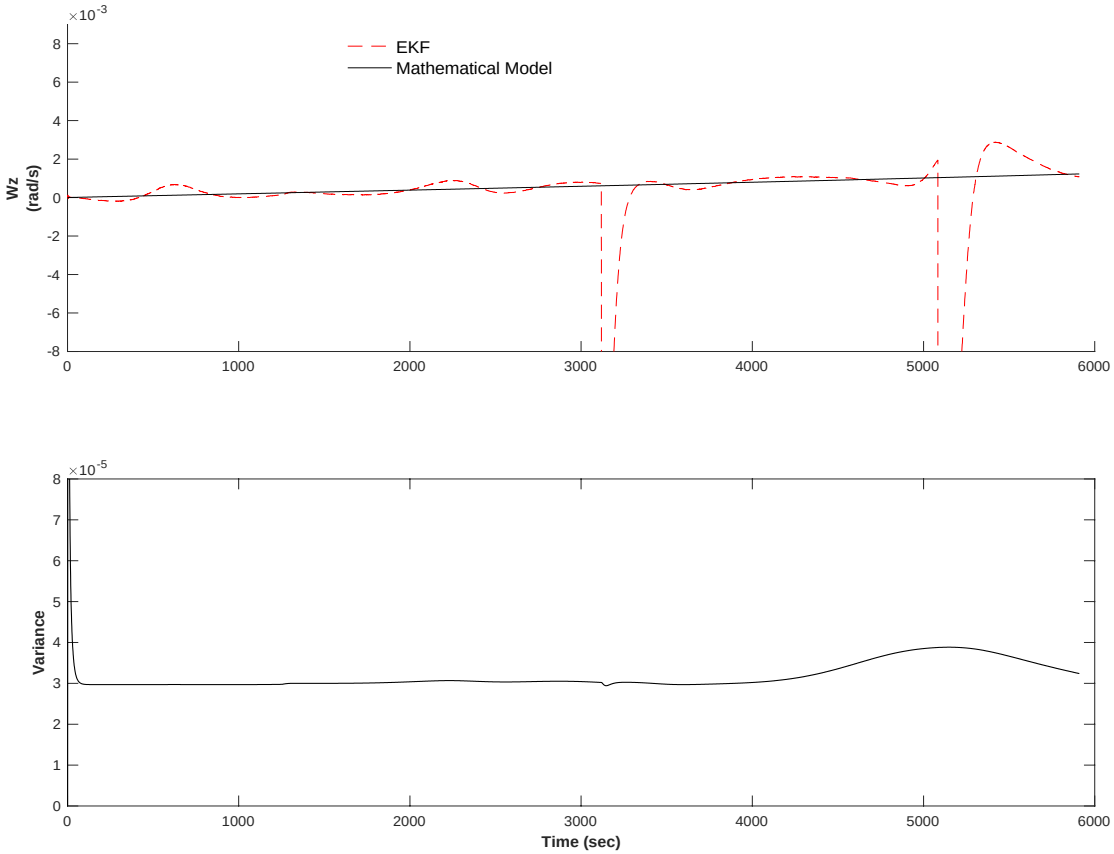

Figure 4.8 : w_z estimation with EKF.

Spike pattern matches with high angle locations on body angle graphs. Deviation from the initial values are small since initial angular velocities are selected low. Second case of the six-state estimation is without magnetometer measurements. Measurement matrix is given below,

$$z = [\phi \quad \theta \quad \psi \quad \omega_x \quad \omega_y \quad \omega_z]^T \tag{4.40}$$

Two cases of different measurement matrices are used to compare EKF's behavior of different number measurements with same states. Body angle and angular velocity graphs are given in Figure 4.9-4.14

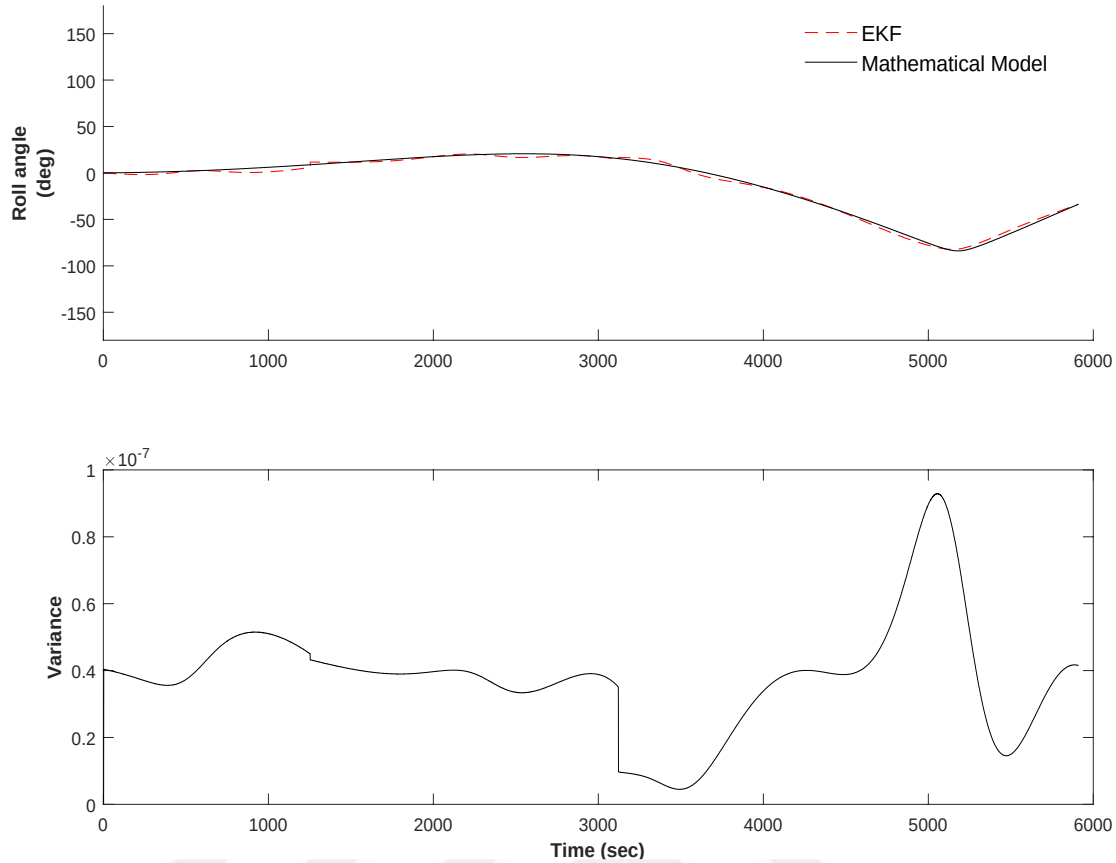

Figure 4.9 : Roll estimation with EKF.

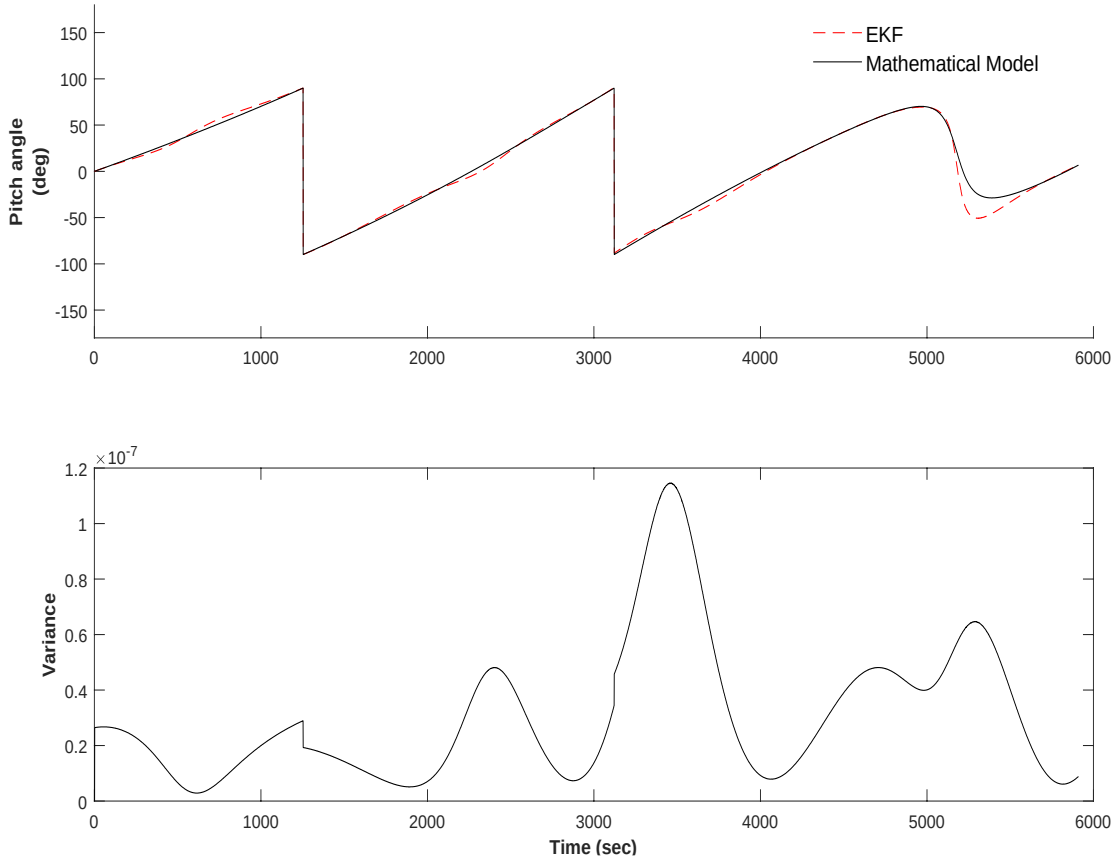

Figure 4.10 : Pitch estimation with EKF.

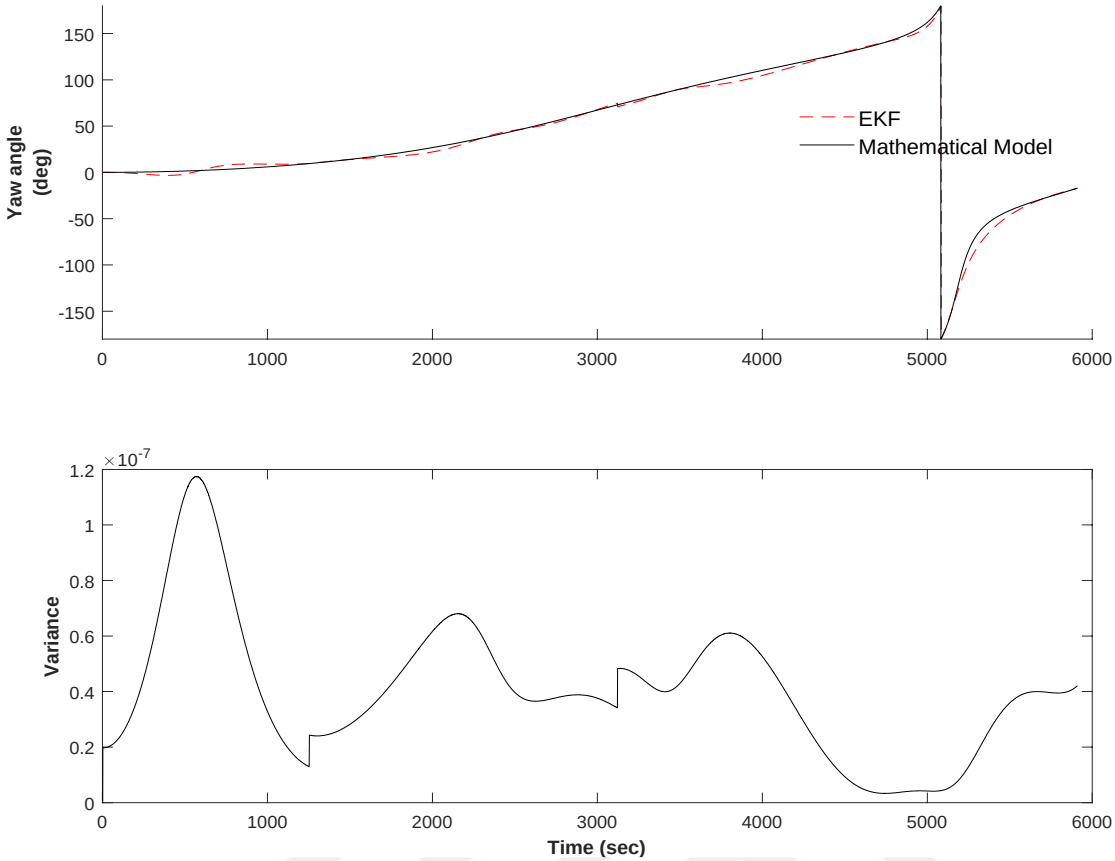

Figure 4.11 : Yaw estimation with EKF.

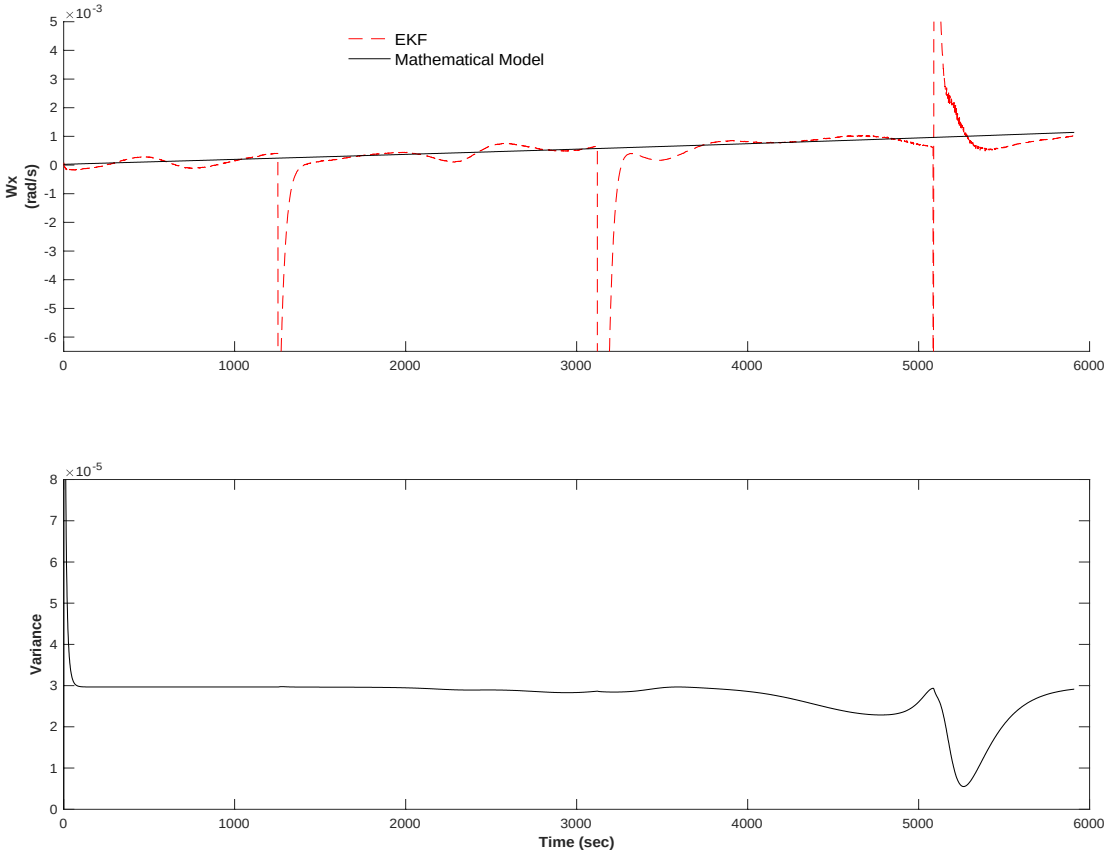

Figure 4.12 : w_x estimation with EKF.

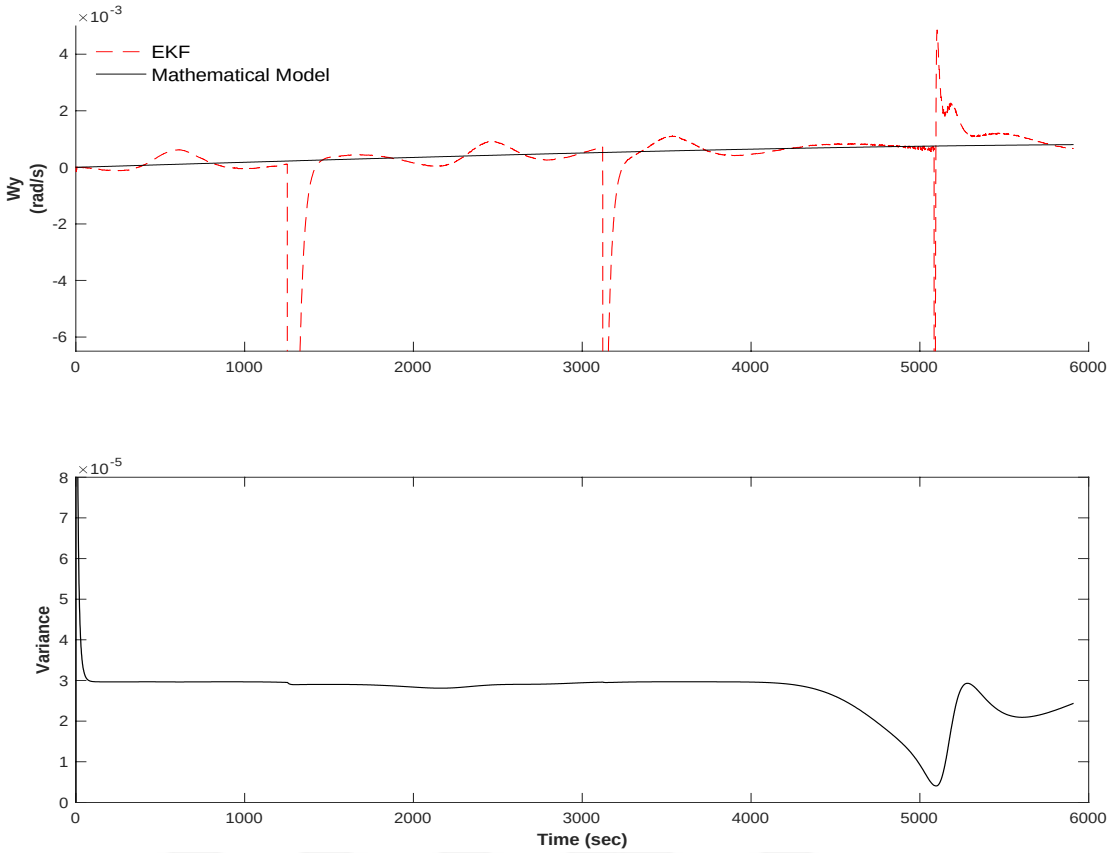

Figure 4.13 : w_y estimation with EKF.

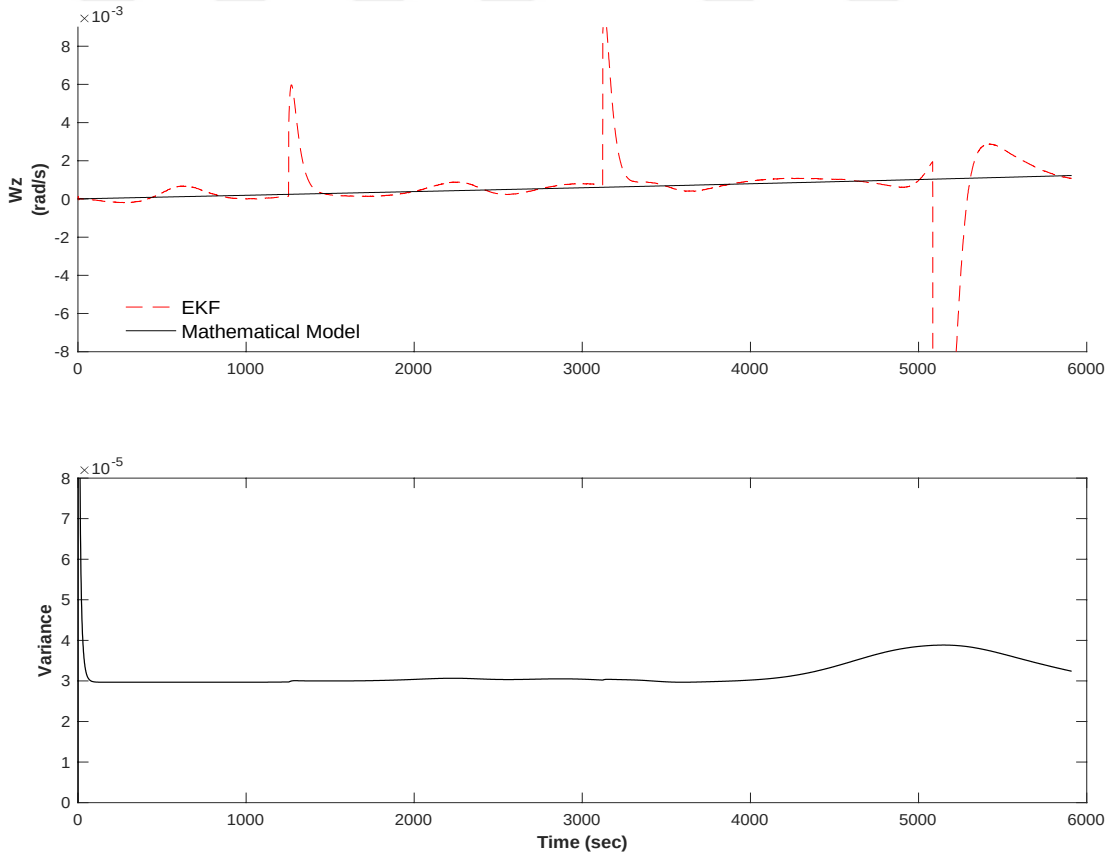

Figure 4.14 : w_z estimation with EKF.

Spike patterns are the same with the first case. A comparion between two cases are given in Table 4.3.

Table 4.3 : EKF RMSE analysis of different measurements.

with Magnetometer measurement			without Magnetometer measurement		
ϕ	θ	ψ	ϕ	θ	ψ
6.0405	6.7231	11.4703	6.0405	6.7878	11.4880

The second estimation includes sensor biases. Three biases for magnetometer and gyros make the second problem a 12-state estimation operation.

$$\mathrm{x} = \begin{bmatrix} \phi & \theta & \psi & \omega_{\mathrm{x}} & \omega_{\mathrm{y}} & \omega_{\mathrm{z}} & \mathrm{b}_{\mathrm{m_x}} & \mathrm{b}_{\mathrm{m_y}} & \mathrm{b}_{\mathrm{m_z}} & \mathrm{b}_{\mathrm{g_x}} & \mathrm{b}_{\mathrm{g_y}} & \mathrm{b}_{\mathrm{g_z}} \end{bmatrix}^{\mathrm{T}} \tag{4.41}$$

Process noise matrix is constructed as,

$$\mathrm{Q} = \begin{bmatrix} 1\mathrm{E}-6I_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 1\mathrm{E}-18I_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 1\mathrm{E}-12I_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 1\mathrm{E}-5I_{3x3} \end{bmatrix} \tag{4.42}$$

The Initial value of the covariance matrix has six additional element and given as $\mathrm{P_o} = \mathrm{diag}\left(\begin{bmatrix} 10^{-1} & 10^{-1} & 10^{-1} & 10^{-3} & 10^{-3} & 10^{-3} & 10^{-4} & 10^{-4} & 10^{-4} & 10^{-4} & 10^{-4} & 10^{-4} \end{bmatrix}\right)^{\mathrm{T}}$.
In Figure 4.15-4.20, body angles and angular velocity estimations are given.

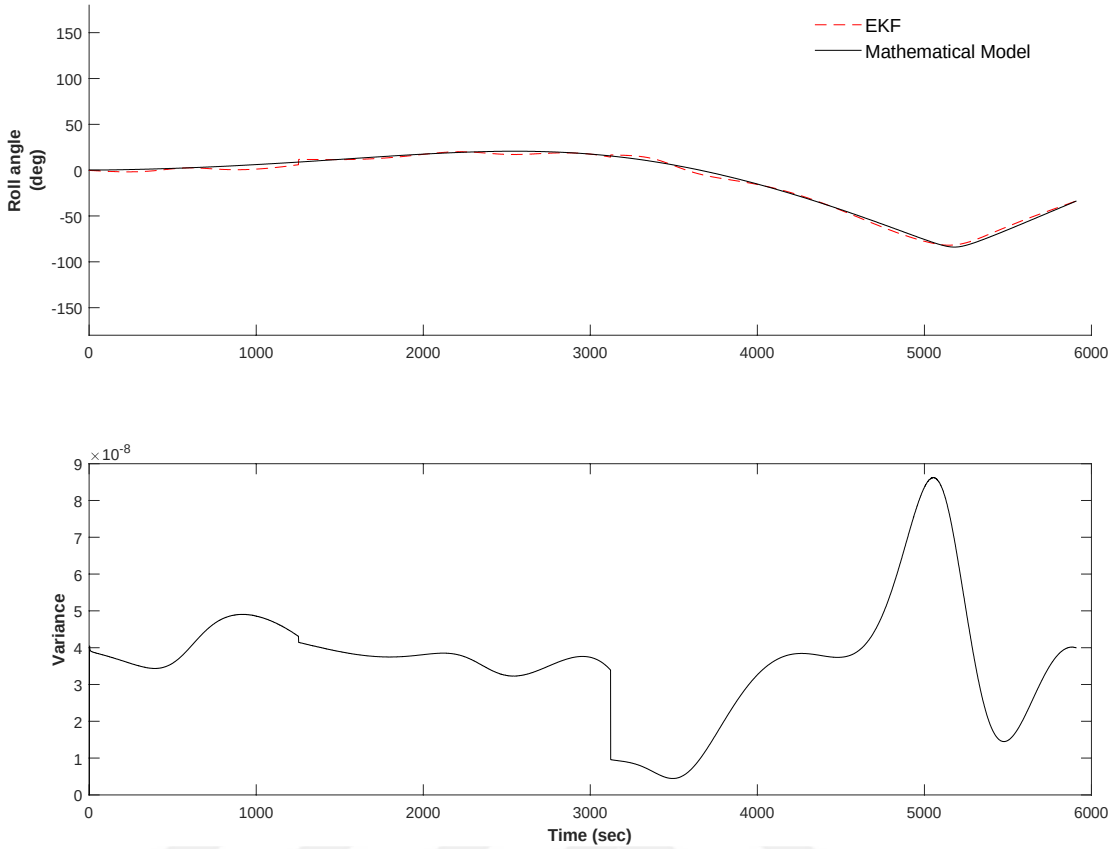

Figure 4.15 : Roll estimation with EKF.

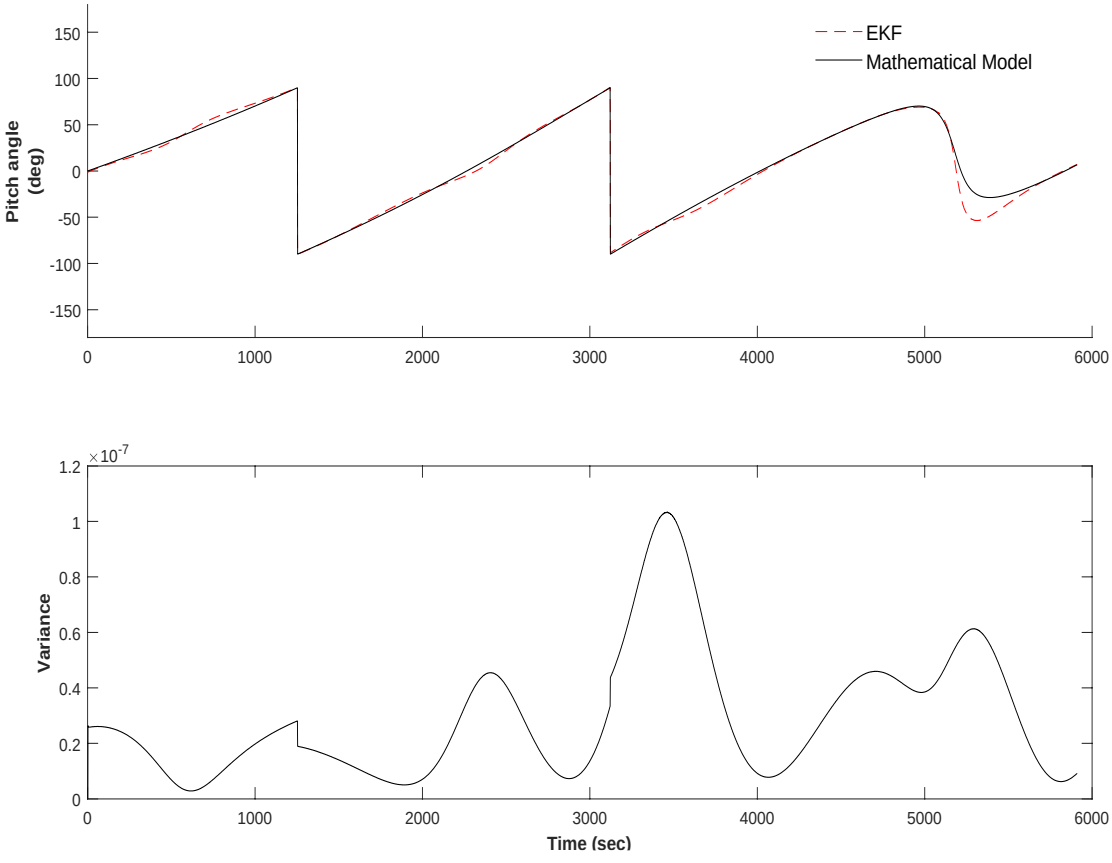

Figure 4.16 : Pitch estimation with EKF.

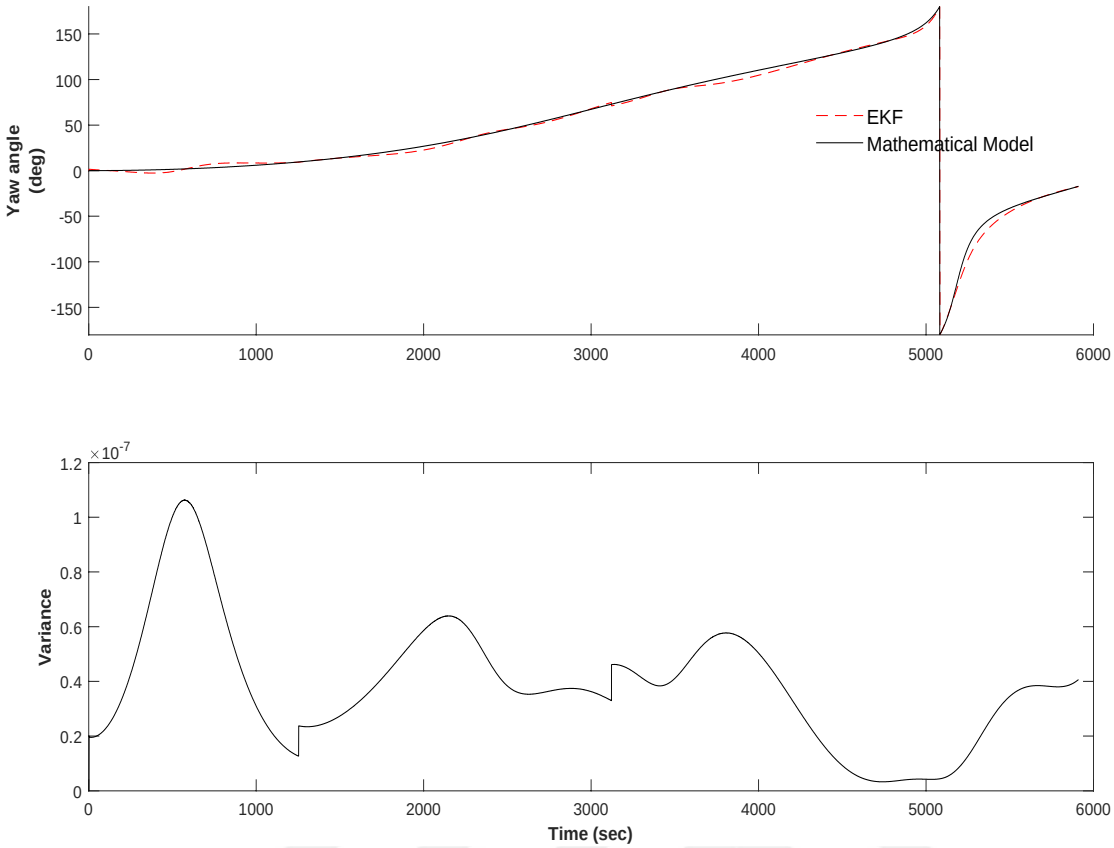

Figure 4.17 : Yaw estimation with EKF.

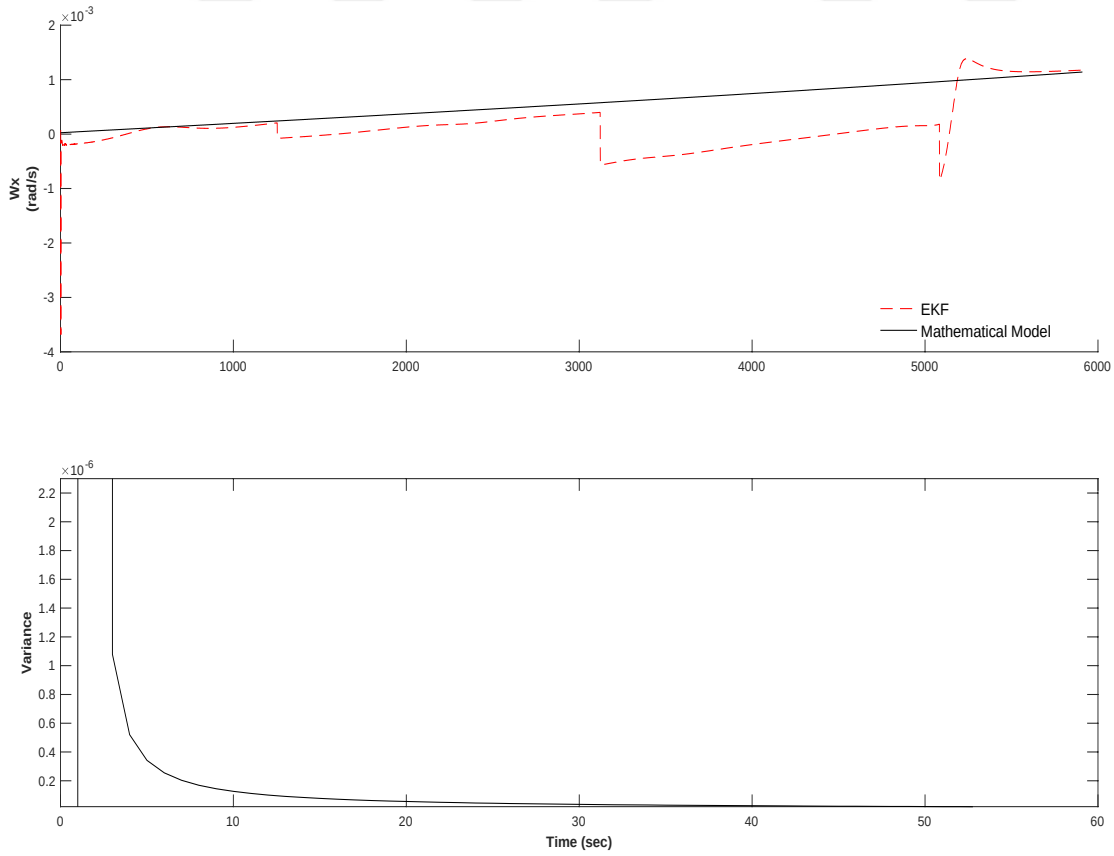

Figure 4.18 : w_x estimation with EKF.

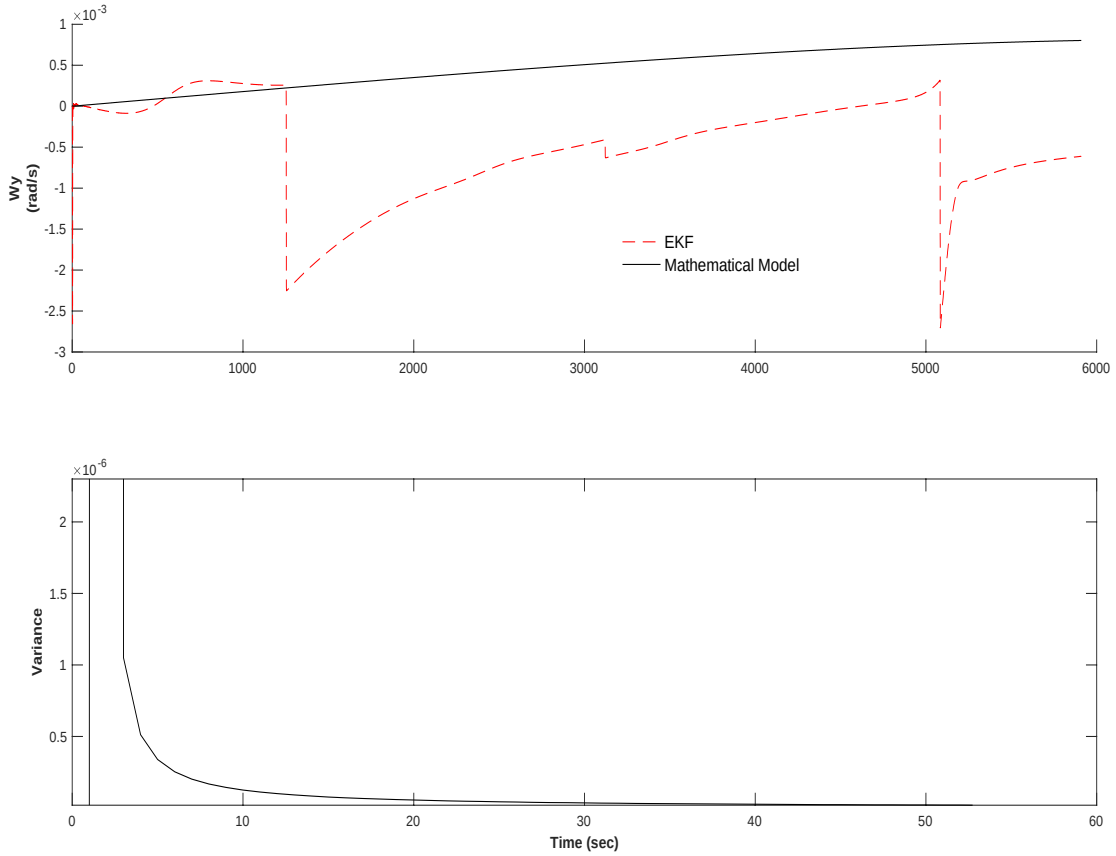

Figure 4.19 : w_y estimation with EKF.

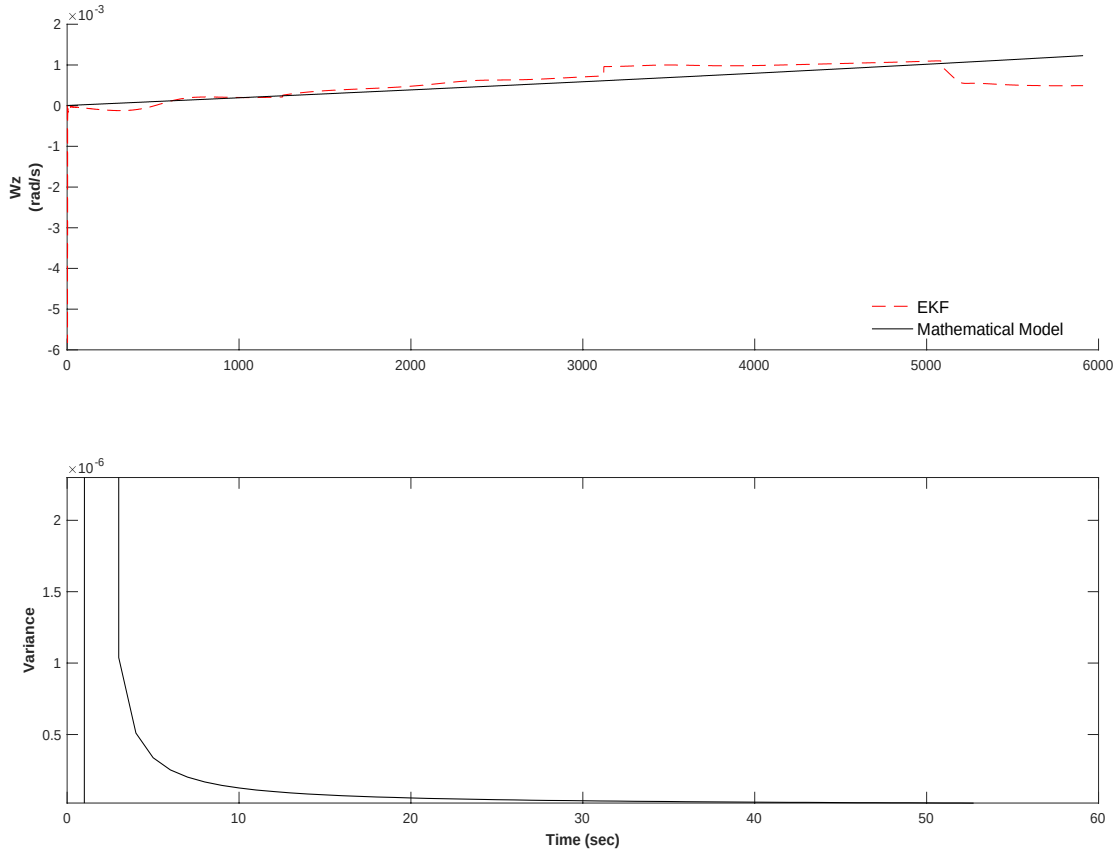

Figure 4.20 : w_z estimation with EKF.

Increasing state size may degragade the Kalman filter performance if measurement numbers stay same. But in this case, apart from minor increases in variances, there is not significant performace loss. In Figures 4.21-4.22, magnetometer and gyro bias estimates are shown.

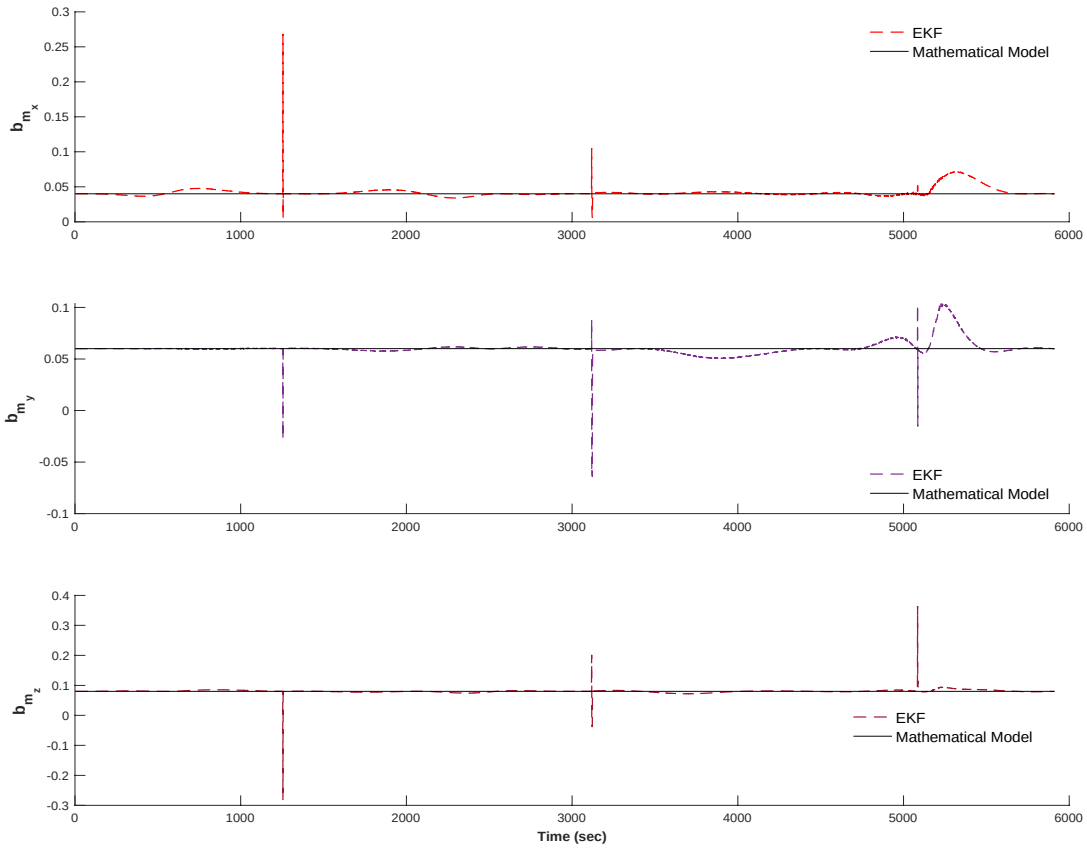

Figure 4.21 : Magnetometer bias estimation with EKF.

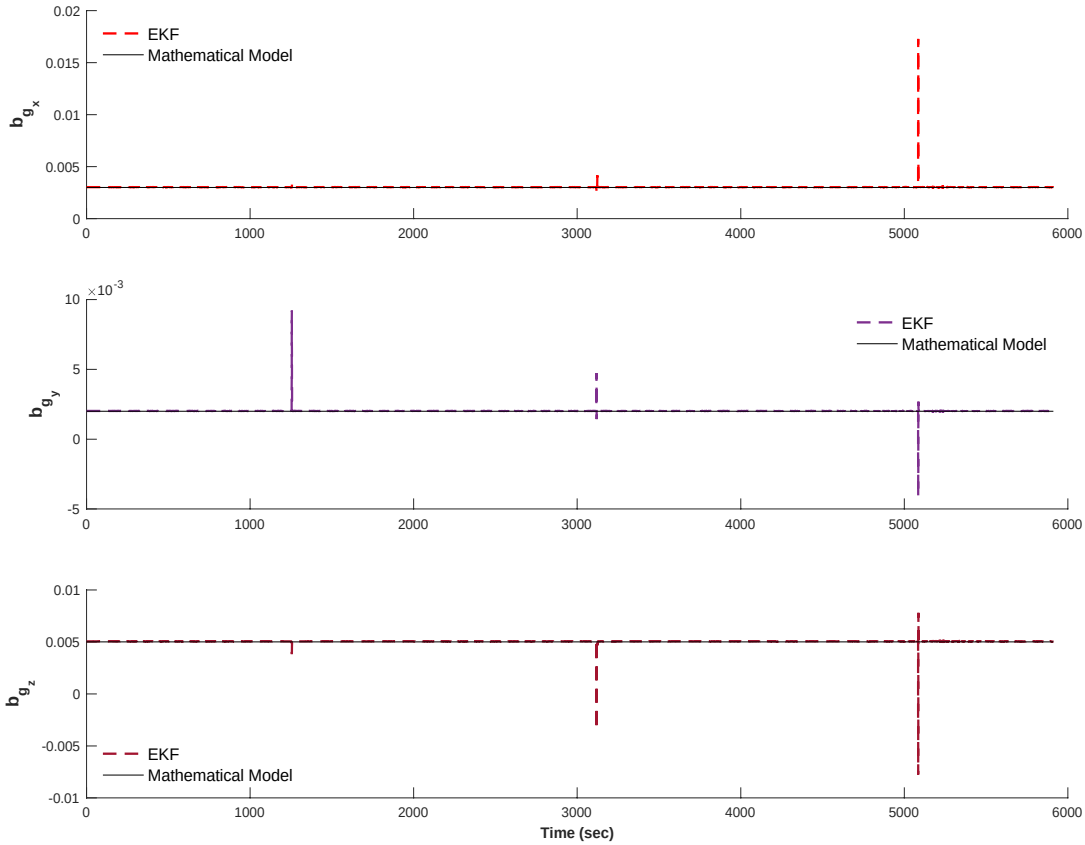

Figure 4.22 : Gyro bias estimation with EKF.

In both cases Kalman filter estimated both biases with minor deviations from the model. Minor spikes are occurred where high pitch and yaw angles are present. In overall case, EKF increased accuracy of the estimation of body angles and estimated angular velocities and sensor biases. Tuning of the Q matrix is important. For bias cases, too high values may cause serious fluctuations. Especially in the angular velocity cases, there is a trade off between RMSE values and sensor output fluctuations. Associated values from the Q matrix are lowered in order to make the sensor output have less fluctuations but it increases error level in RMSE analysis. RMSE analysis of the 12-state EKF estimation is given in Table 4.4

Table 4.4 : EKF RMSE analysis.

EKF 12-state		
ϕ	θ	ψ
4.5339	5.4990	10.5363

4.3 The Unscented Kalman Filter

EKF, due to nature of the first degree linearization, is not as accurate for highly linear systems. Jacobians are hard to derive and the linearization needs very short time intervals otherwise filter becomes unstable. But this comes with the computational power gets higher.The main idead behind the UKF is distributions are easier to approximate from nonlinear funtions (Julier, Uhlmann and Durrant-Whyte, 1995). Therefore it introduces "sigma points". With these points, filter removes need for derivation of Jacobian matrix and it is more robust to the initial estimation errors (Crassidis and Markley, 2003).

The UKF starts with the unscented transformation. It is a process for calculating the statistics of a random variable that undergoes a nonlinear transformation (Julier, 2002).Let x be the state matrix. A sigma vector is constructed with size of 2L+1 where L is dimension of state matrix. These 2L+1 sigma points capture the mean and covariance of the nonlinear functions. Sigma point can be obtained by (Wan and Van Der Merwe, 2000)

$$\chi_0 = \mathrm{x} \tag{4.43}$$

$$\chi_\mathrm{i} = \mathrm{x} + \left(\sqrt{\left(\mathrm{L}+\lambda\right)\mathrm{P_k}}\right)_\mathrm{i} \qquad \mathrm{i} = 1,....\mathrm{L} \tag{4.44}$$

$$\chi_\mathrm{i} = \mathrm{x} + \left(\sqrt{\left(\mathrm{L}+\lambda\right)\mathrm{P_k}}\right)_{\mathrm{i}-\mathrm{L}} \qquad \mathrm{i} = \mathrm{L}+1,....2\mathrm{L} \tag{4.45}$$

χ is denotion for the sigma vector.In order to capture the true nature of the reflection, sigma points are weighted.

$$\mathrm{W}_0^{(\mathrm{m})} = \frac{\lambda}{\left(\mathrm{L}+\lambda\right)} \tag{4.46}$$

$$\mathrm{W}_0^{(\mathrm{c})} = \frac{\lambda}{\left(\mathrm{L}+\lambda\right)} + \left(1-\alpha^2+\beta\right) \tag{4.47}$$

$$\mathrm{W}_\mathrm{i}^{(\mathrm{m})} = \mathrm{W}_\mathrm{i}^{(\mathrm{c})} \tag{4.48}$$

where $\lambda = \alpha^2\left(\mathrm{L}+\kappa\right)-\mathrm{L}$ is scaling parameter. κ is secondary scaling parameter. α determines how spread the sigma points are and β is used to incorporate prior knowledge for x. Then, sigma vector is propagated though nonlinear function.

$$Y_i = f\left(\chi_i\right) \quad i = 0,...2L \tag{4.49}$$

It should be noted that, in (4.44) and (4.45) a square root operation is conducted to the covariance matrix. For a square root of a matrix exists, a matrix must be positive semidefinite. This operation can be conducted via Cholesky factorization. It produces a lower triangular matrix and a conjugate transpose of the lower triangular matrix. Using the weights, propagated state matrix can be determined by,

$$\hat{x}_k^- = \sum_{i=0}^{2L} W_i^{(m)} Y_i \tag{4.50}$$

And propagated covariance,

$$P_k^- = \sum_{i=0}^{2L} W_i^{(c)} \left[Y_i - \hat{x}_k^- \right] \left[Y_i - \hat{x}_k^- \right]^T + Q \tag{4.51}$$

where Q is the process noise. Using the propagated sigma points measured, sigma vector can be obtained by using sensor model equations, denoted by H.

$$Z_i = H\left[Y_i \right] \tag{4.52}$$

Obtaining the measurement matrix,

$$\hat{z}_k^- = \sum_{i=0}^{2L} W_i^{(m)} Z_i \tag{4.53}$$

Second phase of UKF is measurement update.It is similar to EKF.There is still need for calculating two covariances, P_{xy} and P_{yy}.Therefore,

$$P_{yy} = \sum_{i=0}^{2L} W_i^{(c)} \left[Z_i - \hat{z}_k^- \right] \left[Z_i - \hat{z}_k^- \right]^T + R \tag{4.54}$$

$$P_{xy} = \sum_{i=0}^{2L} W_i^{(c)} \left[Y_i - \hat{z}_k^- \right] \left[Z_i - \hat{z}_k^- \right]^T \tag{4.55}$$

where R is the measurement noise. Obtaining the kalman gain is similar to EKF. Hence,

$$K = P_{xy} P_{yy}^{-1} \tag{4.56}$$

The last part is to update both state and covariance matrix with measuremets.

$$\hat{x}_k = \hat{x}_k^- + K\left(z_k - \hat{z}_k^- \right) \tag{4.57}$$

$$P_k = P_k^- - KP_{yy}K^T \tag{4.58}$$

z_k is the measurement matrix. Thus, state matrix and the covariance matrix have been estimated.

4.3.1 Simulation

UKF captures nonlinearities better than EKF. Like in EKF simulations, there are two estimations with UKF in this study. Initially, a 6-state estimation is conducted and then, 12-state estimation with sensor bias are investigated. 6-state model is given below,

$$x = \begin{bmatrix} \phi & \theta & \psi & \omega_x & \omega_y & \omega_z \end{bmatrix}^T \tag{4.59}$$

Also there are two cases of measurement matrix. Initially, magnetometer measurement are considered. In the second case, magnetometer measurements are discarded in order to compare effect of amount of sensor measurements. Measurement matrix in the first case,

$$z = [\phi \quad \theta \quad \psi \quad \omega_x \quad \omega_y \quad \omega_z \quad B_x \quad B_y \quad B_z]^T \tag{4.60}$$

Design parameters are $\alpha = 1E-3$, $\beta = 2$ and $\kappa = 1$. Main problem of the UKF algorithm is square root of the covariance matrix. Covariance matrices are semi-definite positive by nature. But round-off errors are forcing covariance matrices to become non-SPD. This situation makes Cholesky factorization to fail. To overcome this issue a third-party funtion is used (D'Errico, 2020). This function is based on Higham's work that states nearest symmetric SPD to an arbitrary matrix, C, can be shown with (B+H)/2 where H is the symmetric polar factor of B and $B = (C + C')/2$. (Higham, 1988) At the each loop, function checks the matrix to determine whether or not an operation is needed before Cholesky factorization. The state matrices for both simulations are same with EKF simulations. Q matrix for first estimation is given by,

$$Q = \begin{bmatrix} 1E-6I_{3x3} & 0_{3x3} \\ 0_{3x3} & 1E-11I_{3x3} \end{bmatrix} \tag{4.61}$$

First body angle estimation with 13 sigma points are given in Figure 4.23-4.25. Initial covariance matrix is given as $P_o = \mathrm{diag}\left(\begin{bmatrix} 10^{-1} & 10^{-1} & 10^{-1} & 10^{-3} & 10^{-3} & 10^{-3} \end{bmatrix}\right)$.

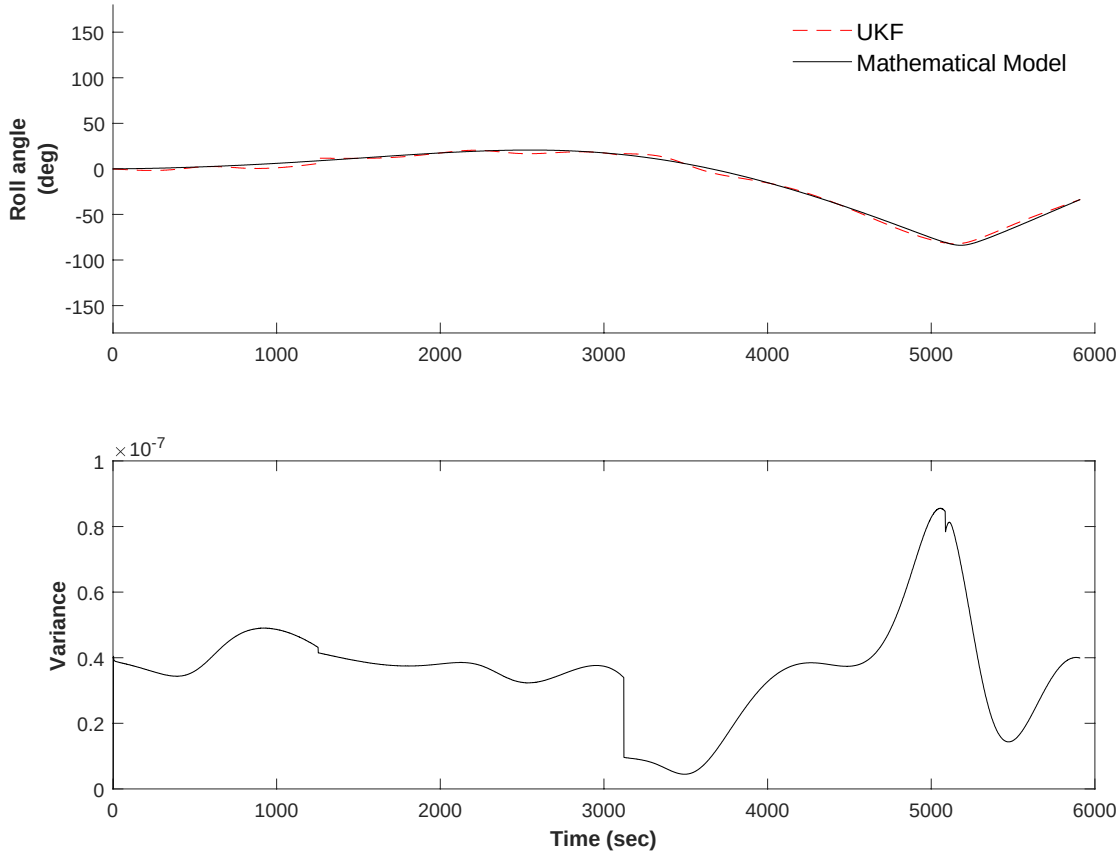

Figure 4.23 : Roll estimation with UKF.

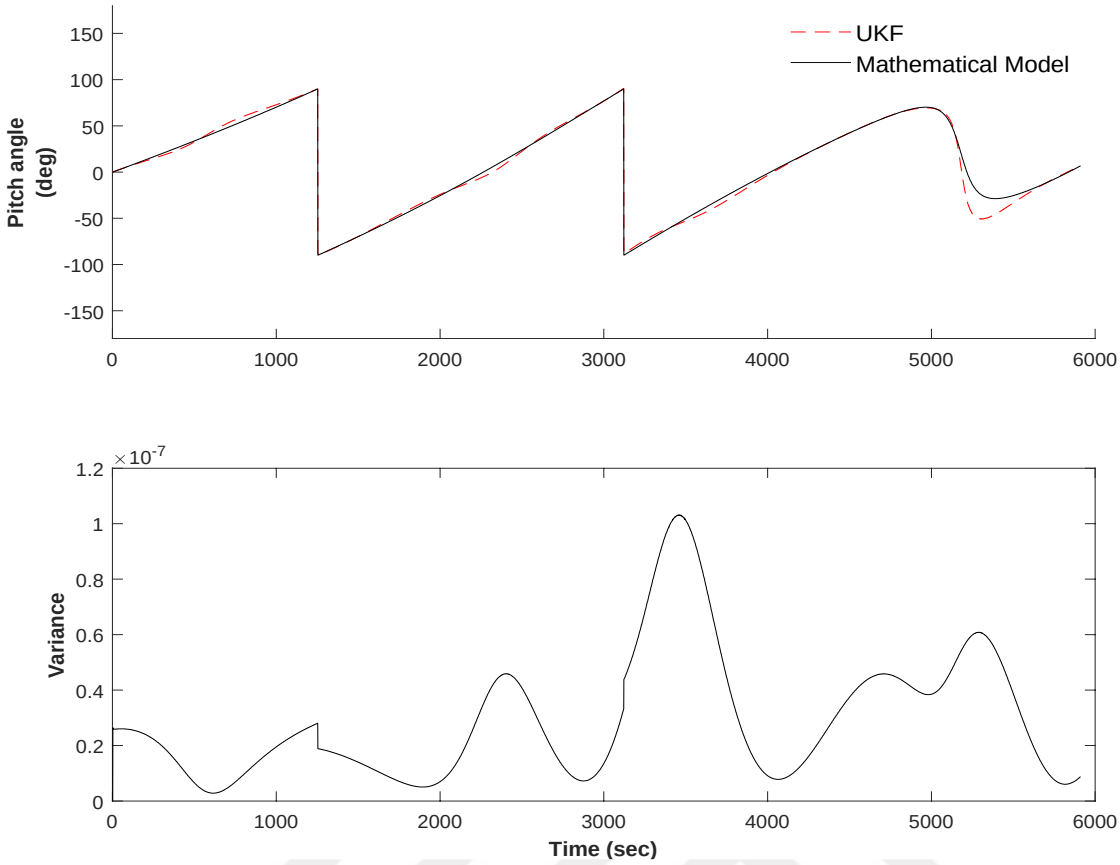

Figure 4.24 : Pitch estimation with UKF.

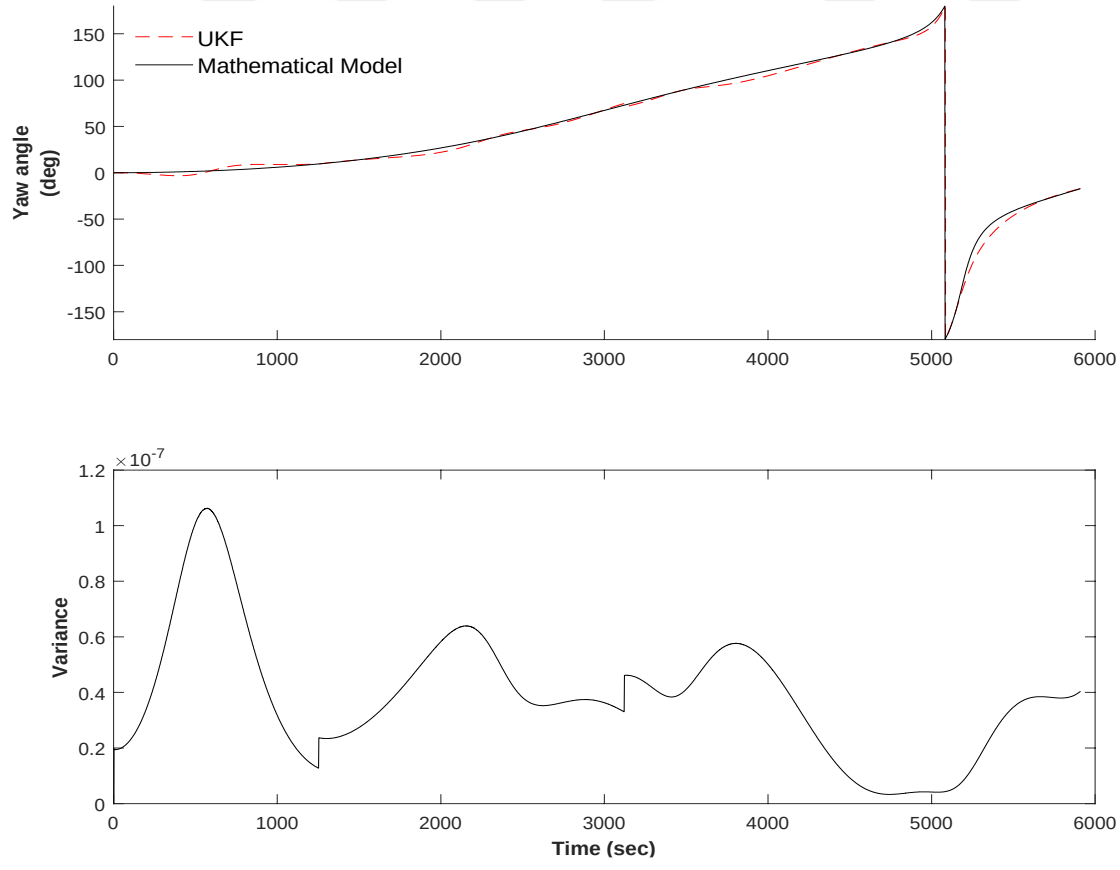

Figure 4.25 : Yaw estimation with UKF.

As can be seen from the figures, high yaw angle caused a spike in the pitch graph. There are no significant difference between UKF and EKF versions for this case. In Figure 4.26-4.28, angular velocity estimations are given.

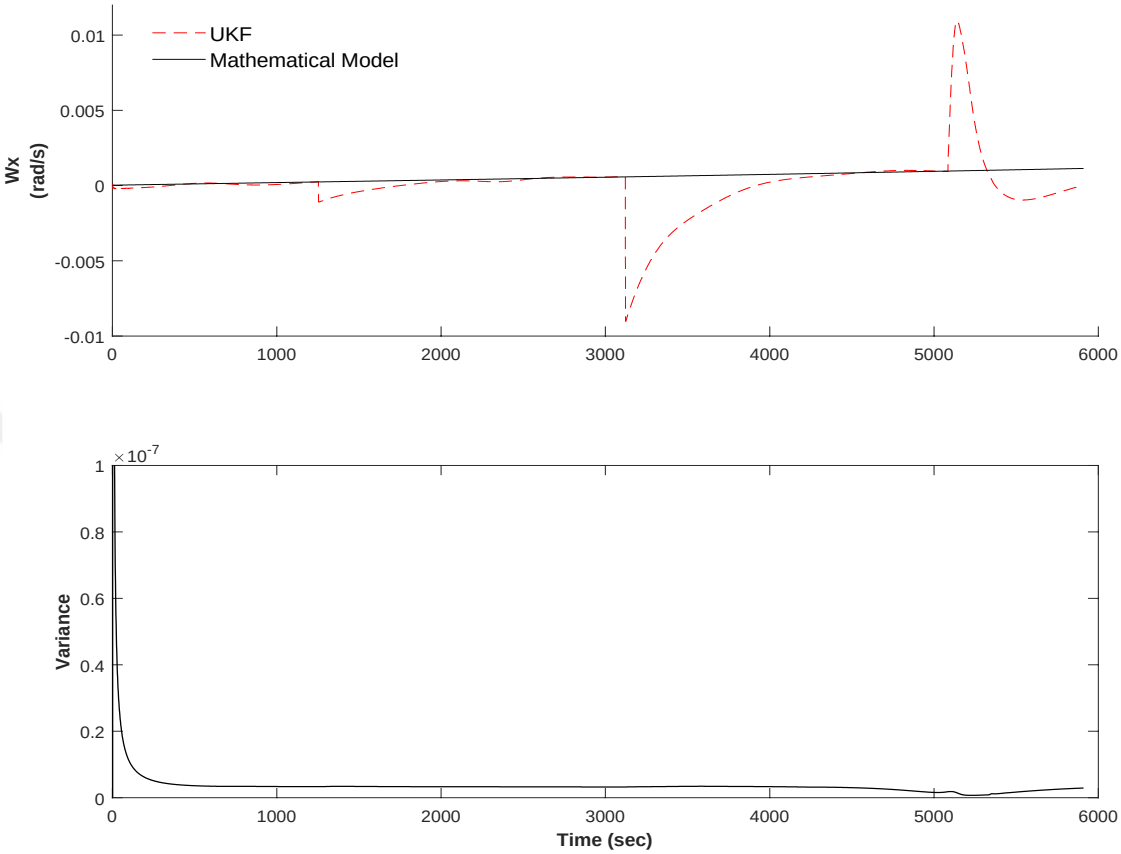

Figure 4.26 : w_x estimation with UKF.

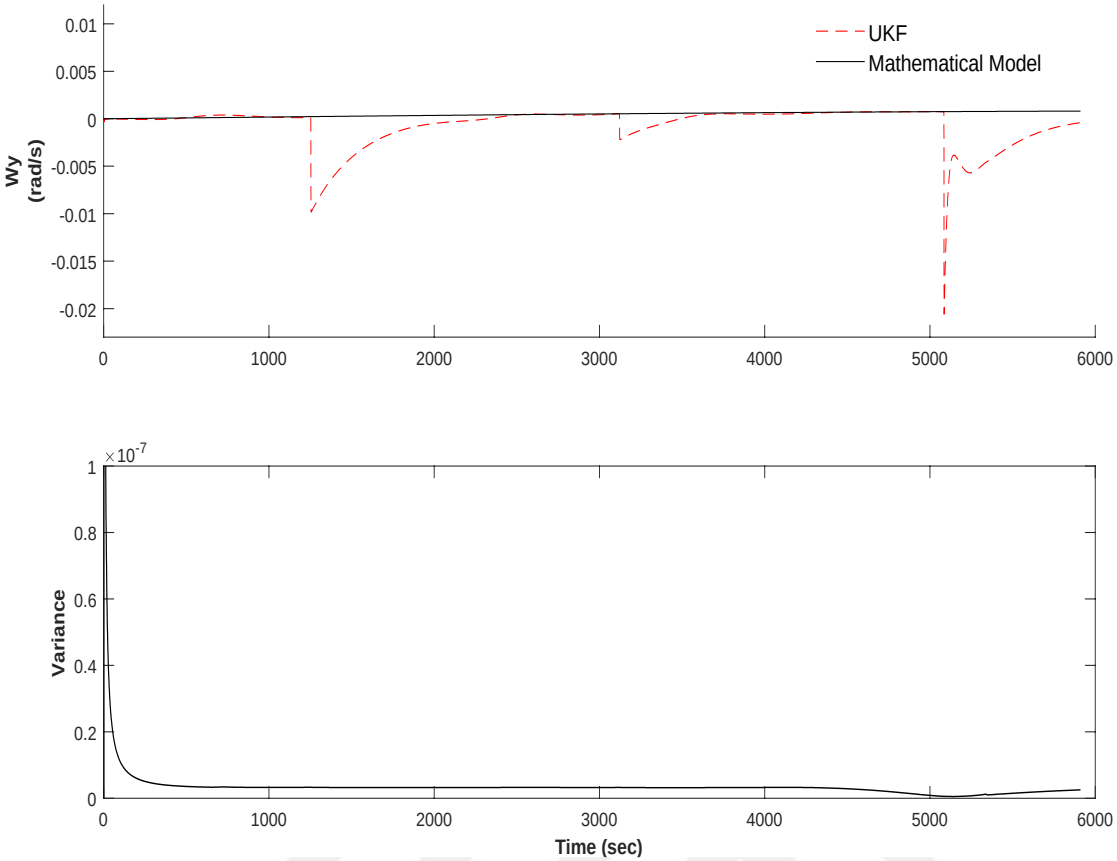

Figure 4.27 : w_y estimation with UKF.

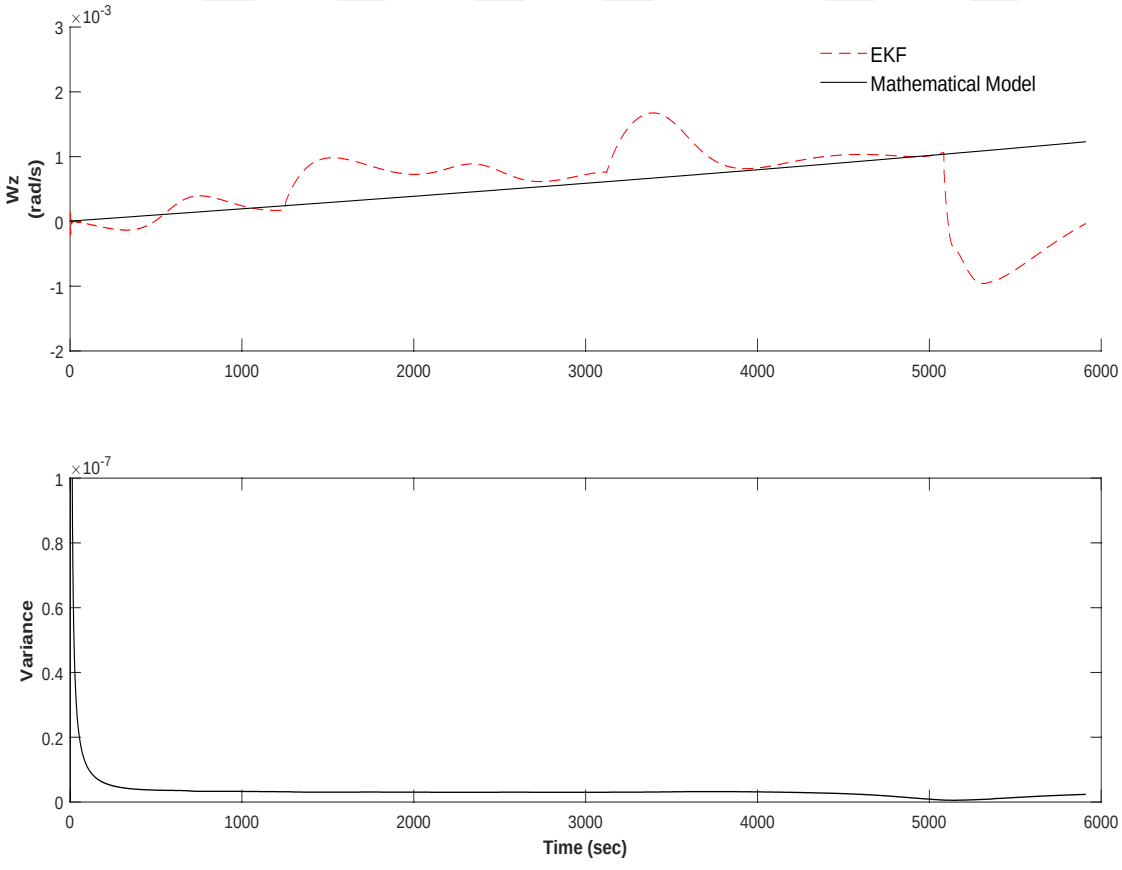

Figure 4.28 : w_z estimation with UKF.

In all of the angular velocity graphs there are noticable spikes. Those patterns are matched with the high yaw and pitch angles. For second case of measurement matrix is constructed without magnetometer measurements. Measurement matrix is given by,

$$z = [\phi \quad \theta \quad \psi \quad \omega_x \quad \omega_y \quad \omega_z]^T \tag{4.62}$$

Body angle and angular velocity graphs are given in Figure 29-34

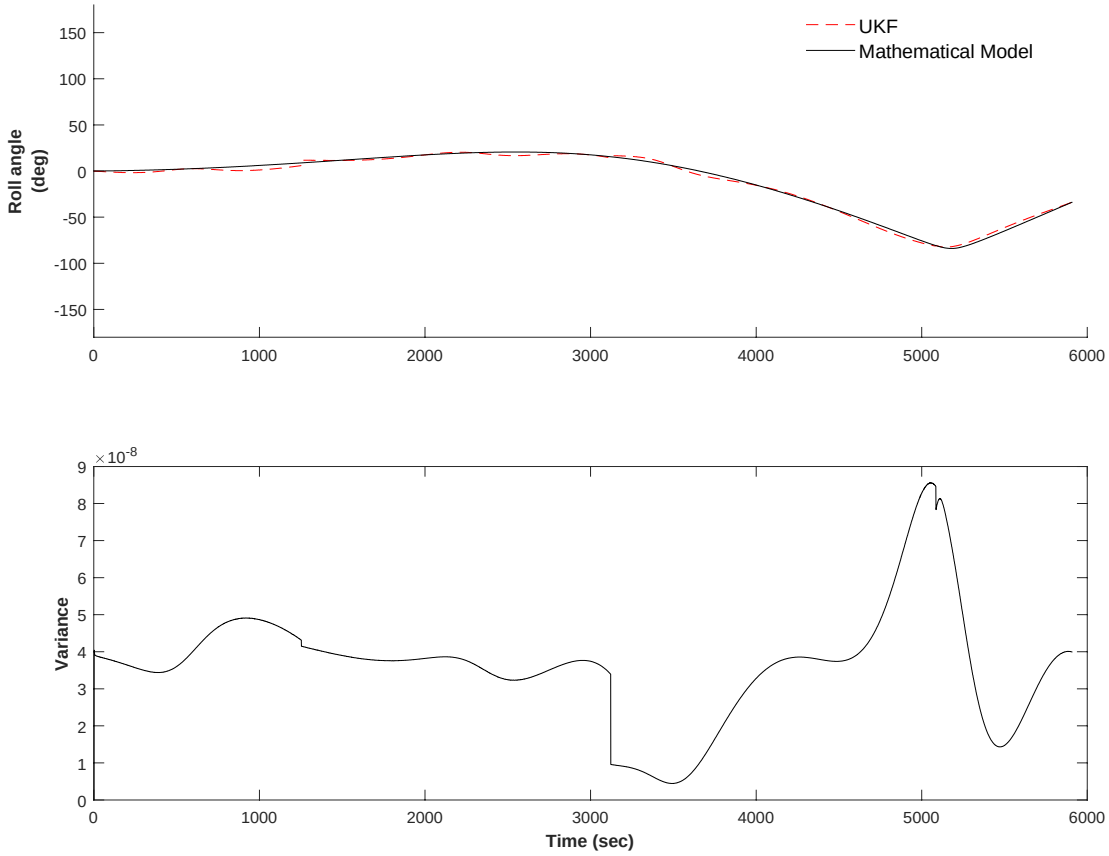

Figure 4.29 : Roll estimation with UKF.

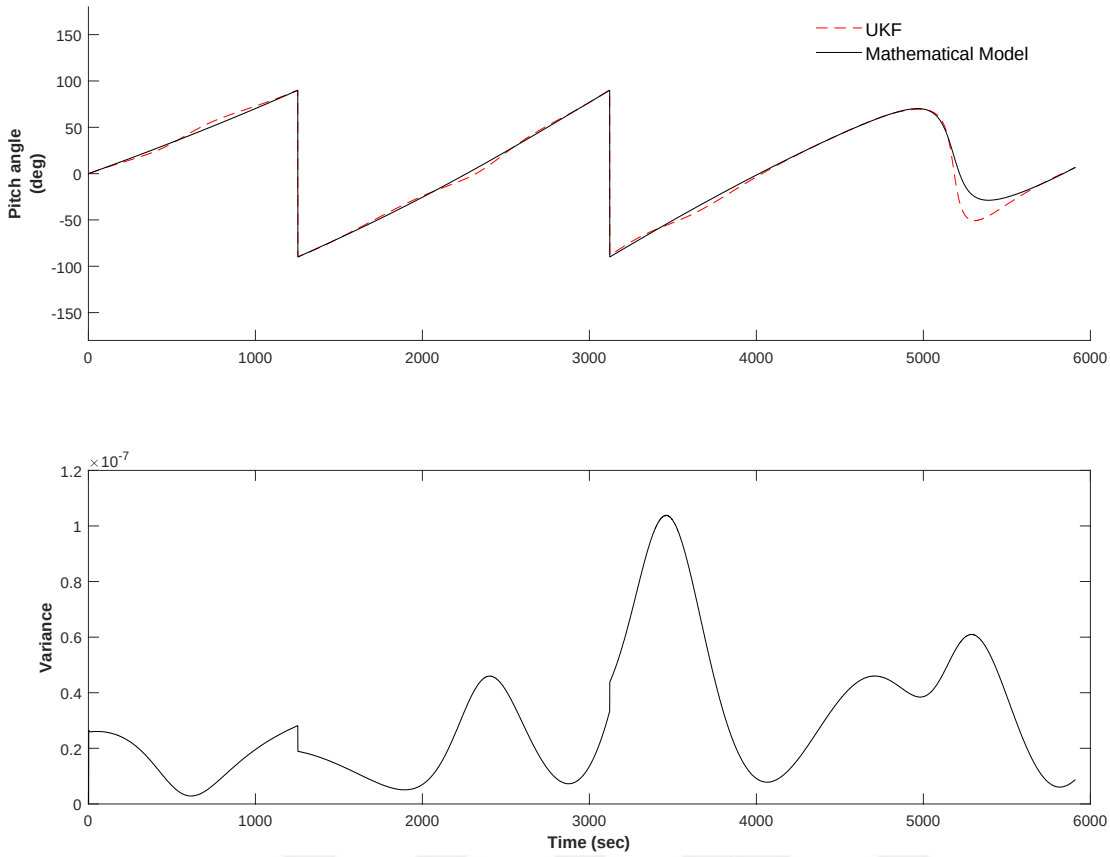

Figure 4.30 : Pitch estimation with UKF.

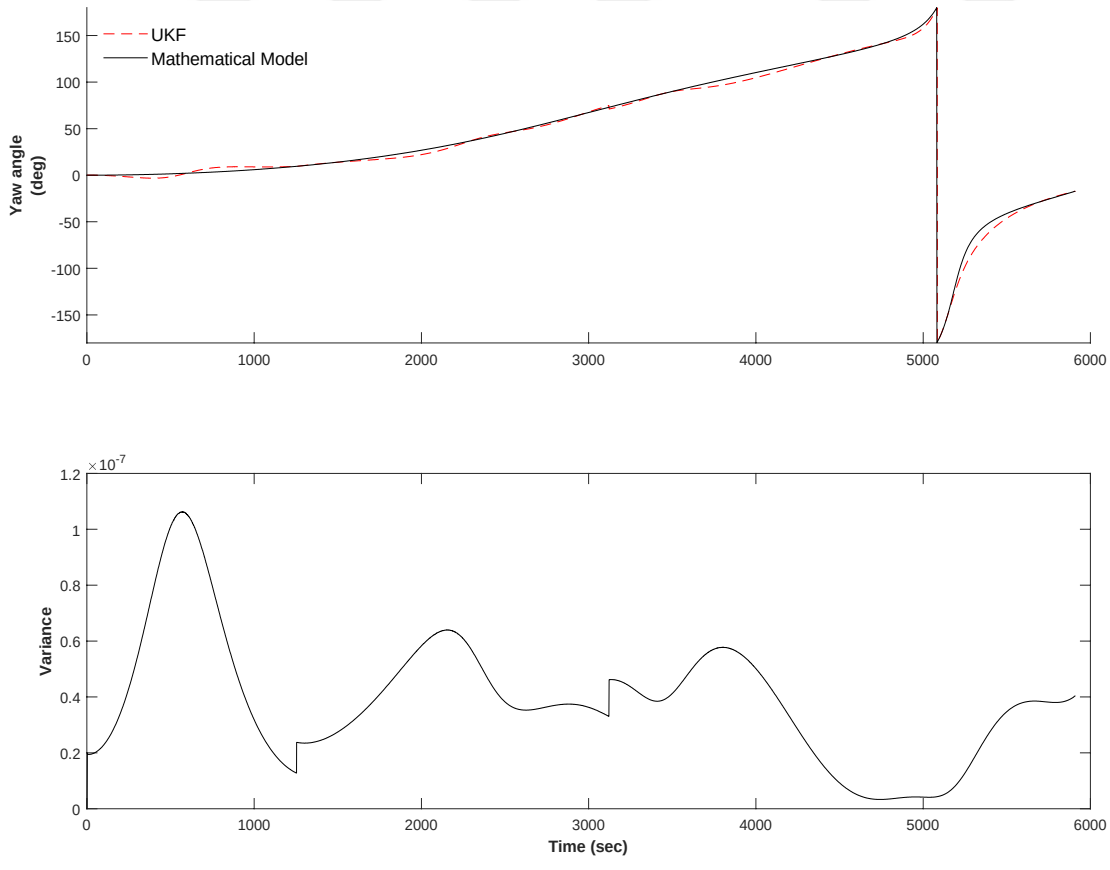

Figure 4.31 : Yaw estimation with UKF.

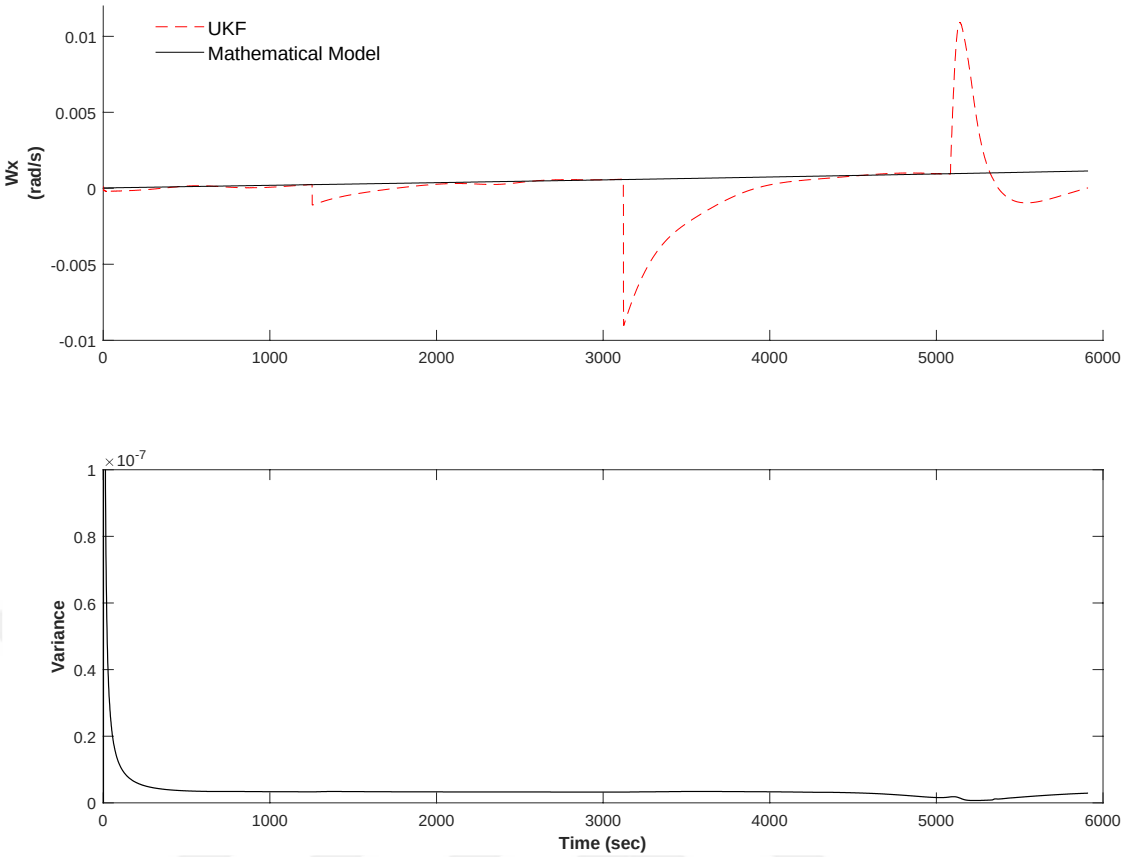

Figure 4.32 : w_x estimation with UKF.

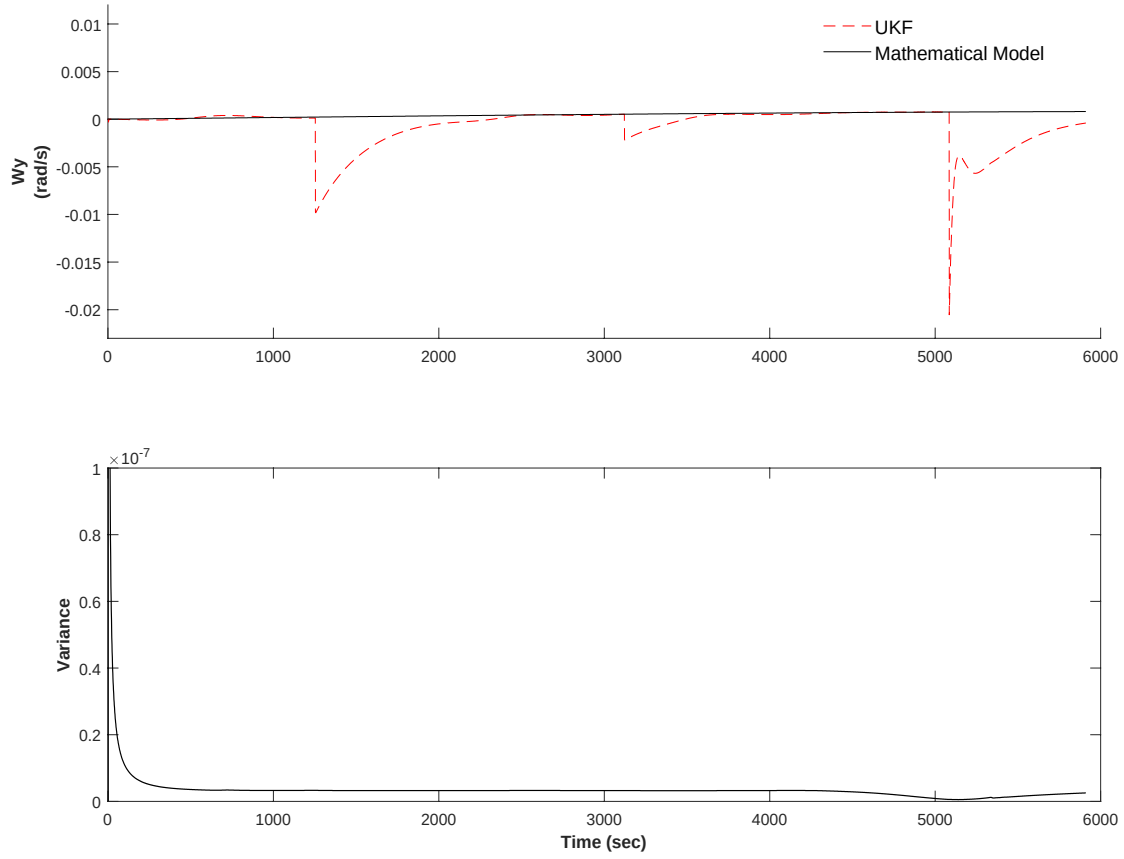

Figure 4.33 : w_y estimation with UKF.

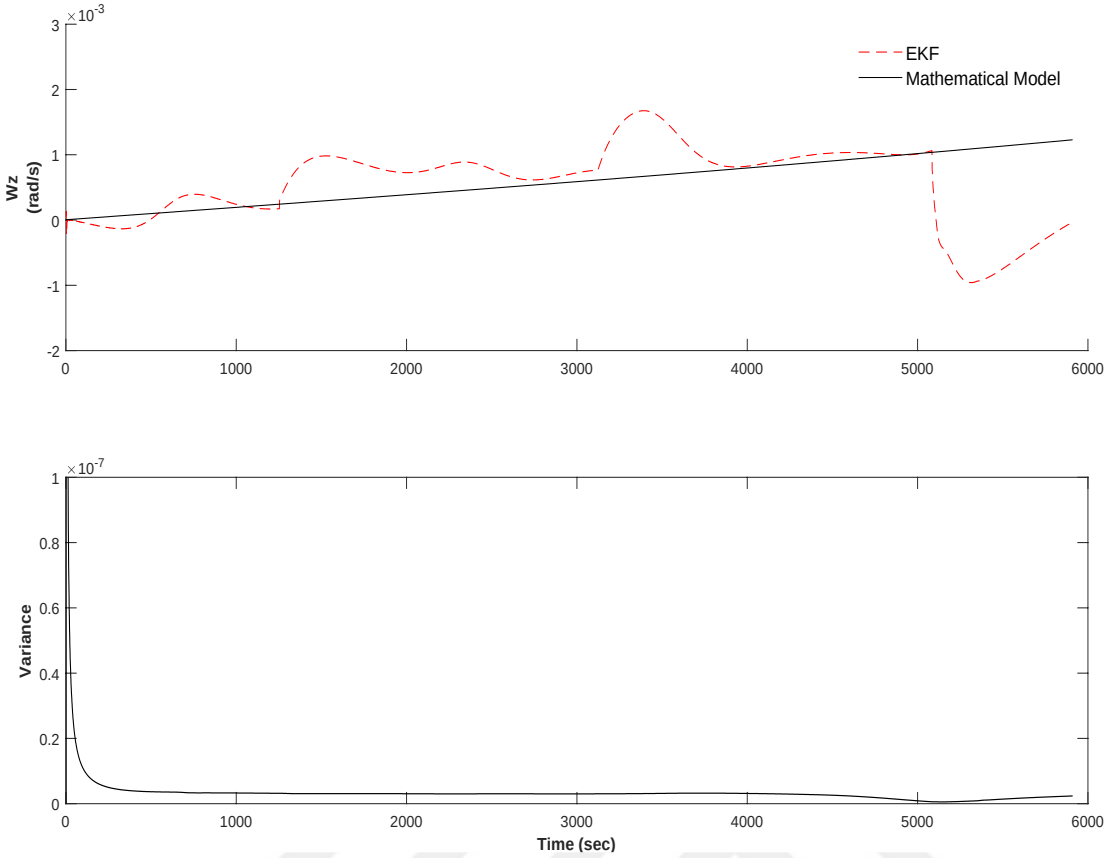

Figure 4.34 : w_z estimation with UKF.

A comparison between two cases is given in Table 4.5.

Table 4.5 : UKF RMSE analysis of different measurements.

with Magnetometer measurement			without Magnetometer measurement		
ϕ	θ	ψ	ϕ	θ	ψ
6.0408	6.7382	11.4697	6.0409	6.8030	11.4848

The second estimation includes sensor biases like in EKF simulations. Three biases for magnetometer and gyros are added to first simulation's measurement matrix.

$$x = \begin{bmatrix} \phi & \theta & \psi & \omega_x & \omega_y & \omega_z & b_{m_x} & b_{m_y} & b_{m_z} & b_{g_x} & b_{g_y} & b_{g_z} \end{bmatrix}^T \quad (4.63)$$

For second simulation, Q matrix is given below.

$$Q = \begin{bmatrix} 1E-6I_{3x3} & 0_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 1E-9I_{3x3} & 0_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 1E-12I_{3x3} & 0_{3x3} \\ 0_{3x3} & 0_{3x3} & 0_{3x3} & 1E-5I_{3x3} \end{bmatrix} \quad (4.64)$$

The Initial value of the covariance matrix has six additional element and given as $P_o = \mathrm{diag}\left(\begin{bmatrix} 10^{-1} & 10^{-1} & 10^{-1} & 10^{-3} & 10^{-3} & 10^{-3} & 10^{-3} & 10^{-3} & 10^{-3} & 10^{-3} & 10^{-3} & 10^{-3} \end{bmatrix}\right)^T$.

Second simulation results given in Figure 4.35-40

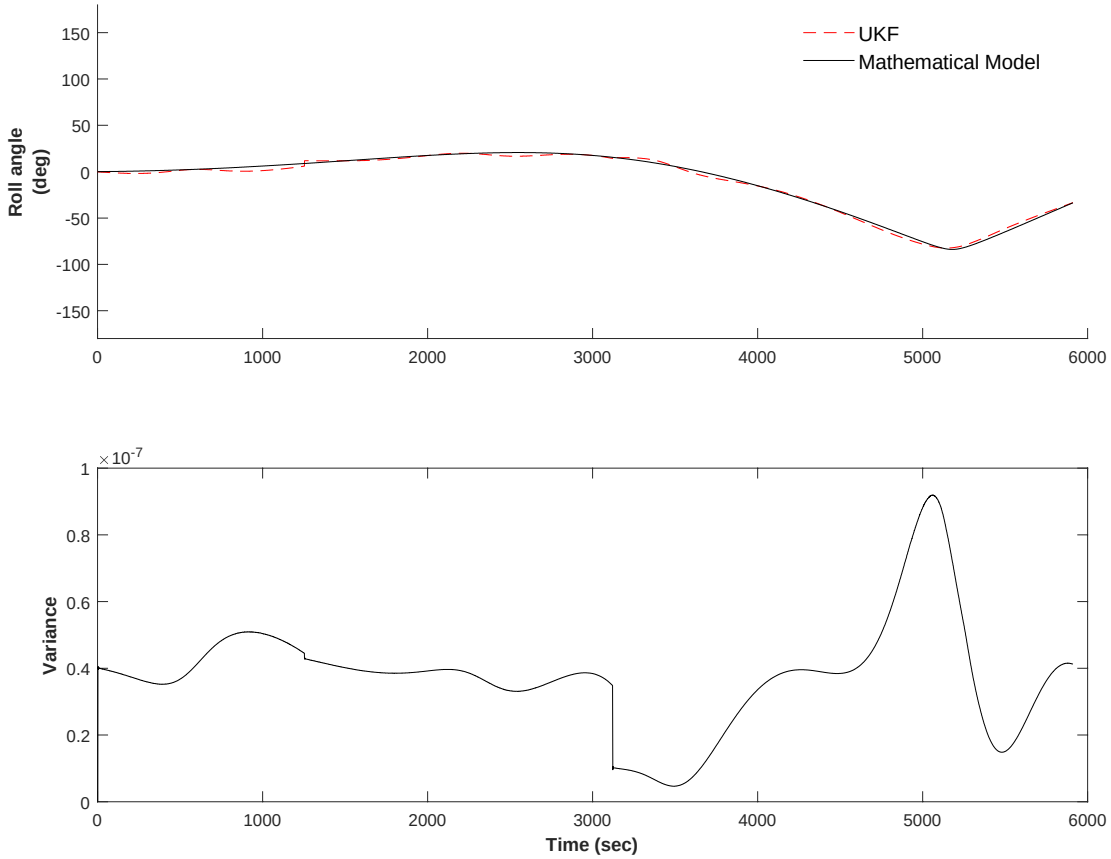

Figure 4.35 : Roll estimation with UKF.

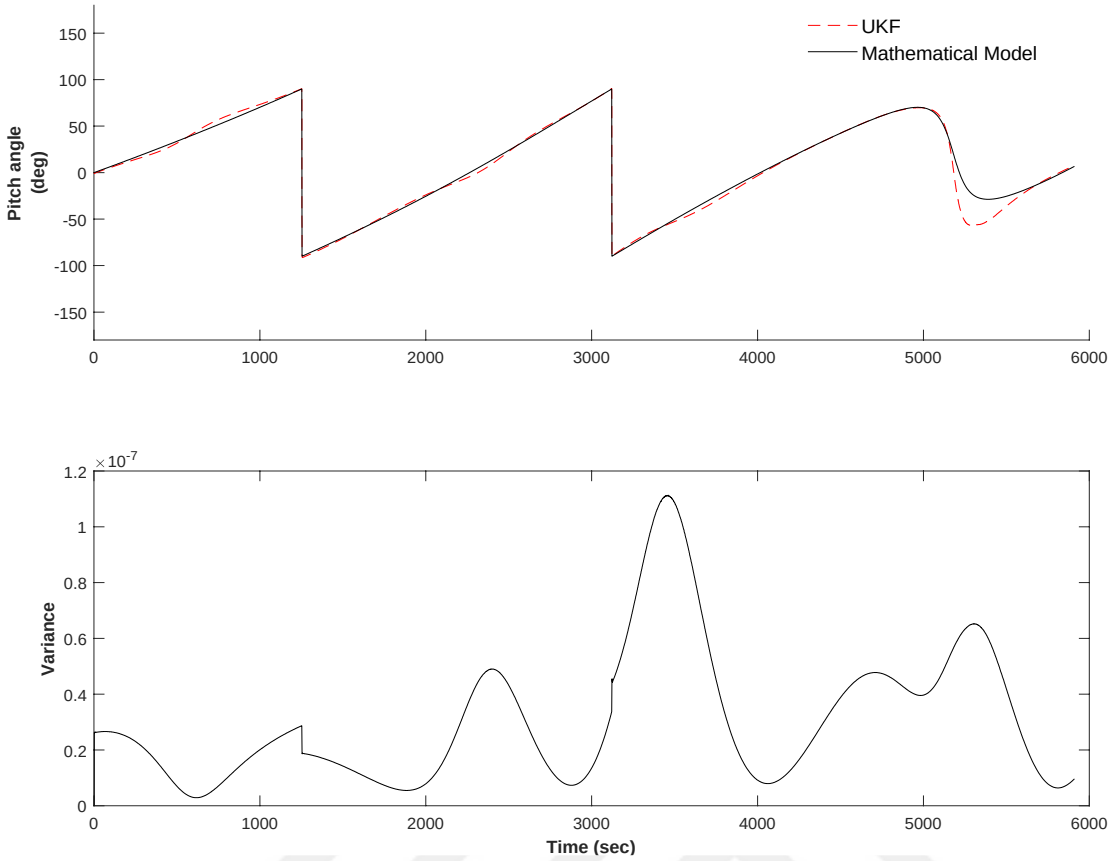

Figure 4.36 : Pitch estimation with UKF.

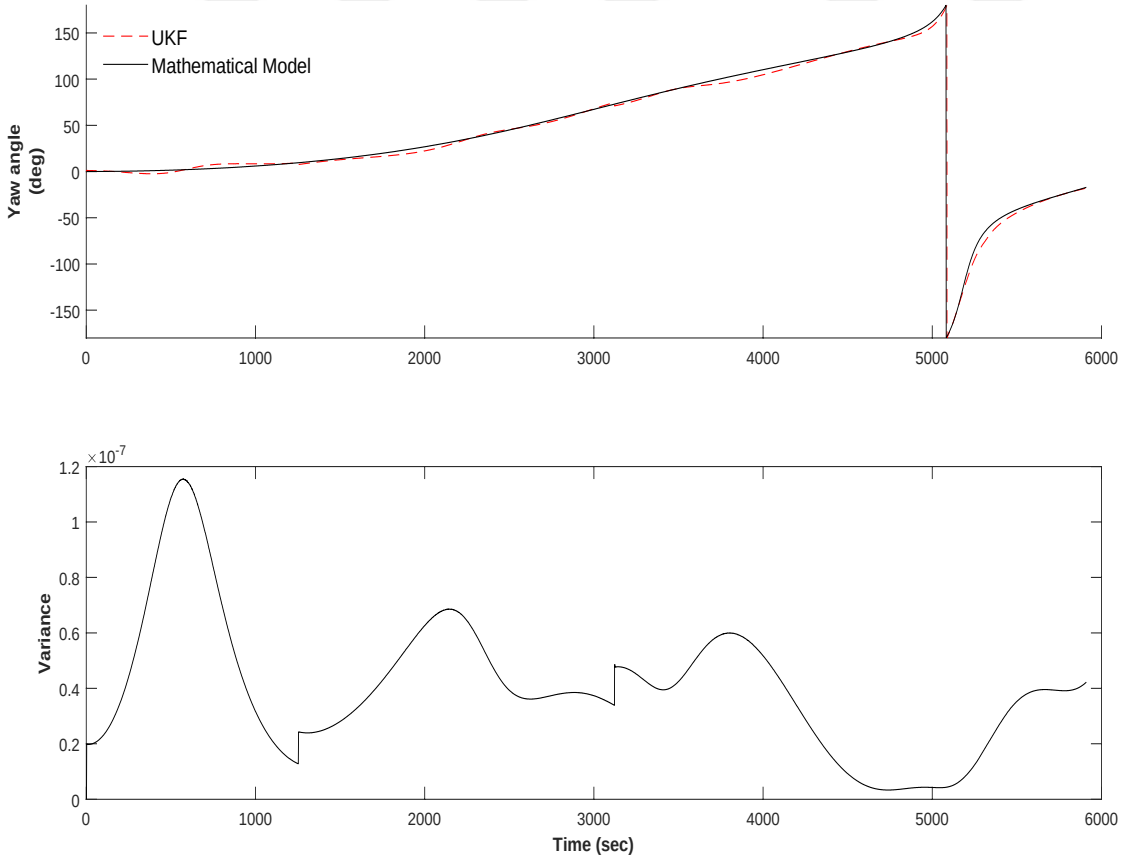

Figure 4.37 : Yaw estimation with UKF.

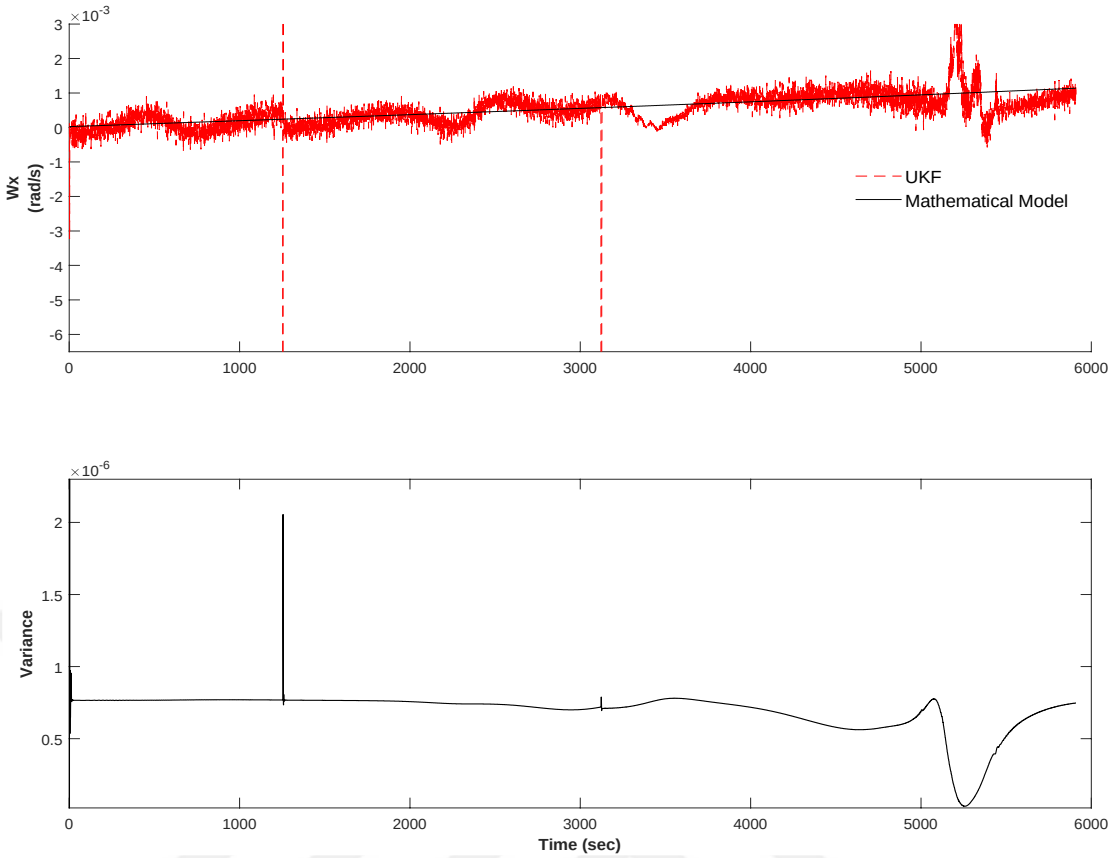

Figure 4.38 : w_x estimation with UKF.

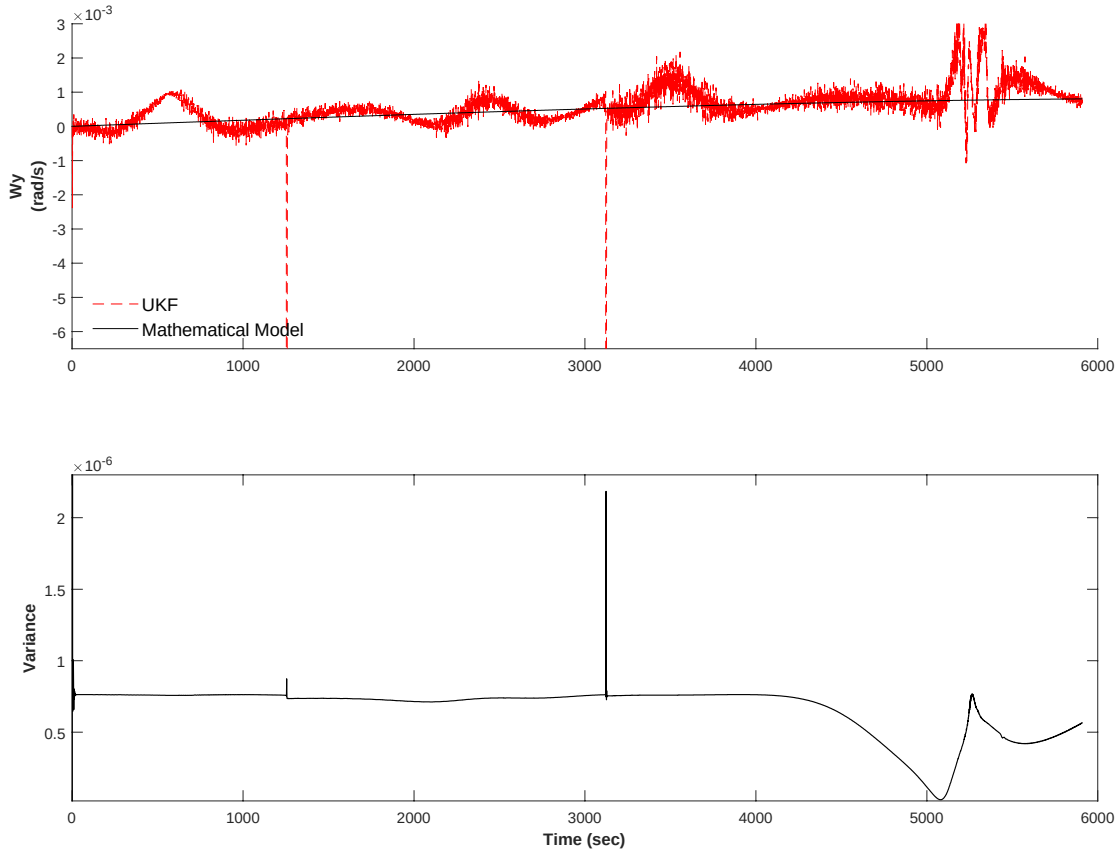

Figure 4.39 : w_y estimation with UKF.

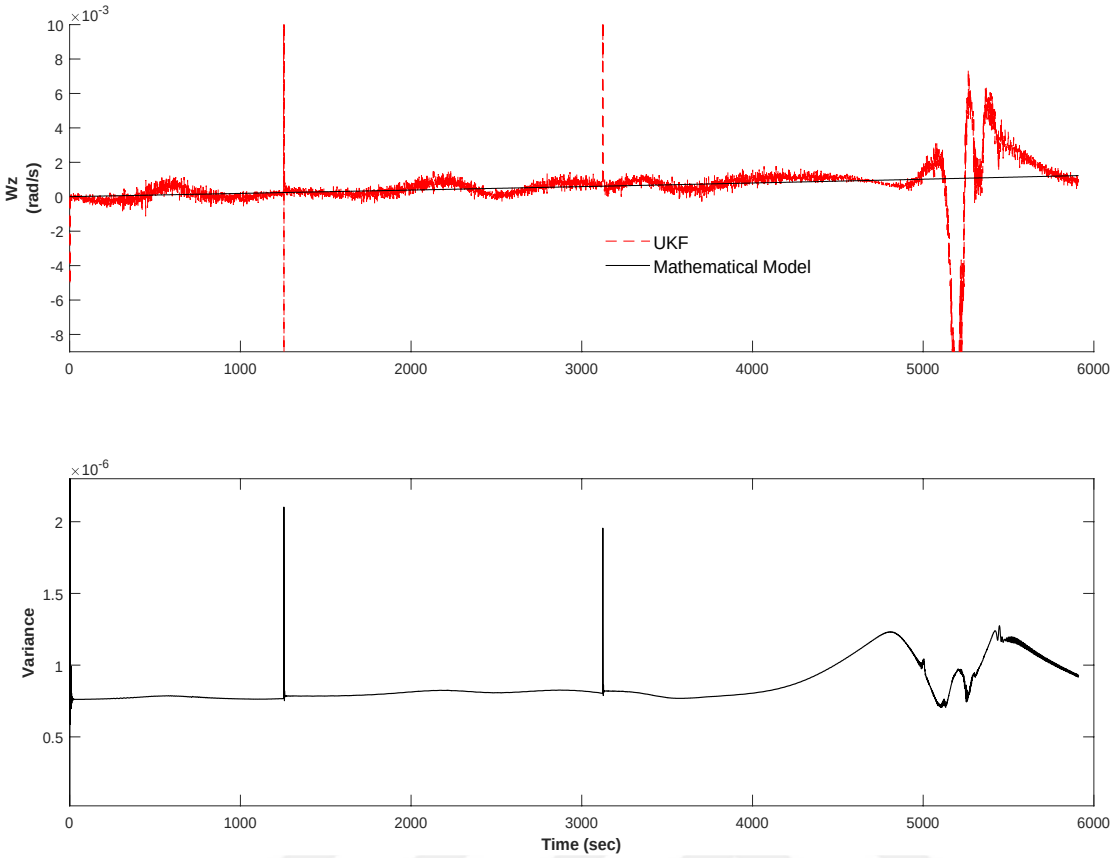

Figure 4.40 : w_z estimation with UKF.

Body angle estimation are without any major spikes. UKF estimated the pitch curve at the end of the time period better than EKF. Angular velocity results are showing that filter estimated all of the velocities well even though in some steps it suffered from high body angles. In this case lowering the Q matrix values caused instabilities. In Figure 4.41-4.42, bias estimations are presented.

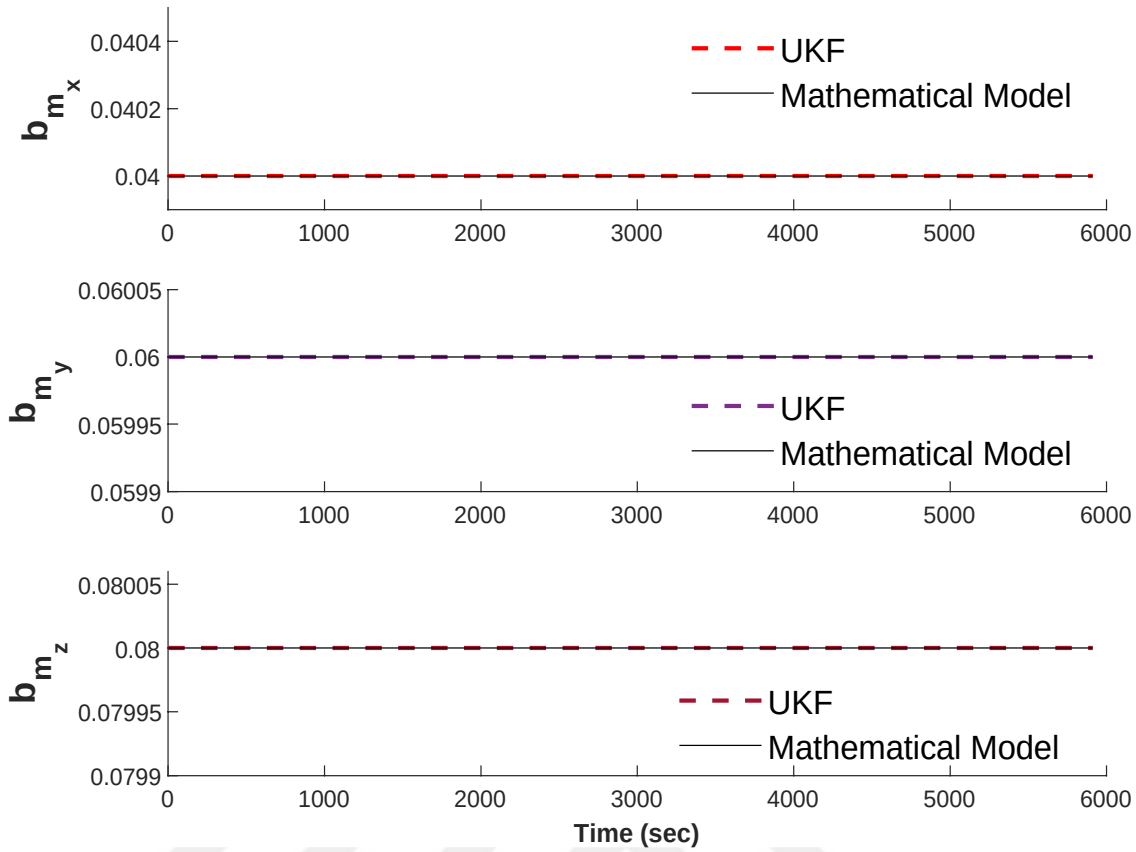

Figure 4.41 : Magnetometer bias estimation with UKF.

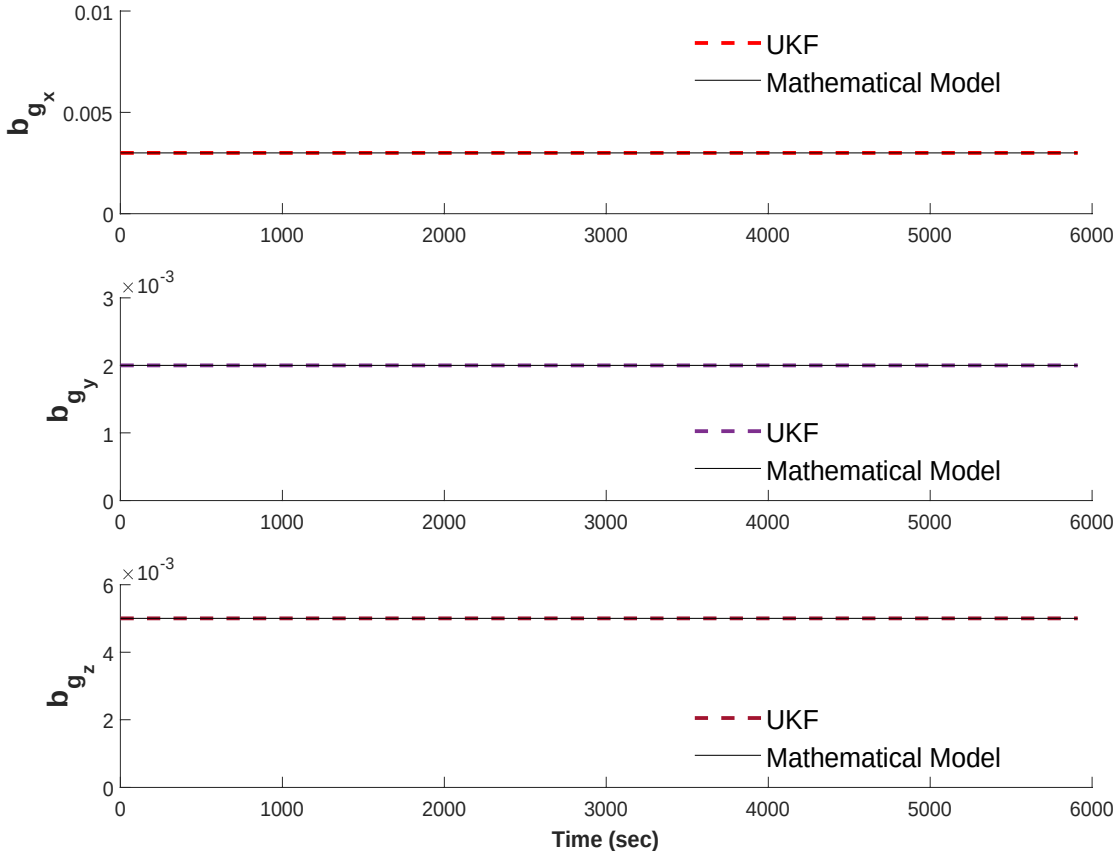

Figure 4.42 : Gyro bias estimation with UKF.

Both of the biases are estimated very close to mathematical model. There are some deviations from the model but magnitude of them are very low that they can be ignored. RMSE analysis of 12-state UKF estimation is given in Table 4.6

Table 4.6 : UKF RMSE analysis.

UKF 12-state		
ϕ	θ	ψ
3.4718	6.1306	10.7370

4.4 Adaptive Filtering

Dynamic and measurement models are not perfect. They contain incomplete information about the actual systems. Even though many sensor models are regarded as accurate enough, systems can still suffer from faulty measurements. In nanosatellite cases, systems are highly interconnected. Robustness of the spacecraft is a top priority since the adverse effects of the space environment can force a mission to be failed. EKF and UKF works with fault-free system accurately. These filter-outputs can diverge very fast if an abnormal value is measured. For these reasons, an adaptation method is required.

The very core idea behind the adaptive methods is to compare real and measurement values to check if the sensor is operating correctly. When there are faulty measurements, systems real error exceeds the its theoretical counterpart. As mentioned above kalman gain decides whether to trust the model or the measurement. For adaptation, kalman gain matrix is multiplied with several factors. Using multiple factors instead of a single factor have its benefits. With multiple factors, a faulty equipment can be specifically targeted. Algorithms can be tuned to discregard the equipment's faulty inputs for that case. (Xia et al., 1994)

Compensate the adverse effects, an adaptive fading kalman filter is used. System covariance matrix, P_k, is assumed to has uncertainties which are caused by incomplete dynamic equations. The estimated covariance matrix, P_k, is related to P_k by ζ_k, forgetting factor. (Kim, Lee and Park, 2006)

$$\mathrm{P}_k = \zeta_k \mathrm{P}_k \tag{4.65}$$

There are several methods for finding the ζ_k. In this work, forgetting factor is taken as zero. It is considered that dynamic equations completely define the satellite dynamics. For faulty measurement, there is a similar approach. Using the innovation covariances,

$$\tilde{C}_k = \alpha_k C_k \tag{4.66}$$

where $\tilde{C}_k$ is estimated innovation covariance. C_k is defined by (4.54) for UKF and (4.24) for EKF. $\tilde{C}_k$ can be written as,

$$\nu\nu^T \tag{4.67}$$

where ν is innovation matrix.The factor matrix, α_k can now be calculated. Using (4.66)

$$\alpha_k = \max\left\{1, \frac{\mathrm{diag}\left(\tilde{C}_k\right)}{\mathrm{diag}\left(C_k\right)}\right\} \tag{4.68}$$

Element wise division of innovation covariances forms the scaling factor. *max* function states that no element of the α_k can be smaller than 1. Since the faulty measurement effect is compansated by decresing the magnitude of the K, elements smaller than one increase the magnitude of K.

$$\tilde{K} = \begin{bmatrix} \frac{K_{11}}{\alpha_1} & \cdots & \frac{K_{1N}}{\alpha_N} \\ \vdots & \ddots & \vdots \\ \frac{K_{M1}}{\alpha_1} & \cdots & \frac{K_{MN}}{\alpha_N} \end{bmatrix} \tag{4.69}$$

Each column of the kalman gain is divided by its matching scaling factor row. The adaptive filter only runs when the sensor data is faulty. At the other time steps, filters use expected algorithms.Filter needs to sense that sensor measurements are corrupted. Using a statistical function (Hacijev and Soken, 2013)

$$\hbar(k) = \nu(k+1)^T \left[P_{yy}\right]^{-1} \nu(k+1) \tag{4.70}$$

This function has chi-square (χ^2) distribution. Its degree of freedom is equal to innovation vector's dimension. If the level of significance is defined by, η

$$\mathrm{U}\left\{\chi^2 > \chi^2_{\eta,\mathrm{p}}\right\} = \eta, \quad 0 < \eta < 1 \tag{4.71}$$

a threshold value can be obtained. If there is a malfunction in the system, $\hbar(\mathrm{k})$ will be bigger than $\chi^2_{\eta,\mathrm{p}}$.

4.4.1 Simulation

Series of simulations were conducted with both UKF and EKF in order to test the effect of multiple faulty measurements. In this case, there are two different disturbance addition to the system, bias failure and noise increment. Both cases are tested with different amount of faulty sensor data. UKF and EKF's response to faulty measurements are compared with adaptive filters. In both filter cases, simulations were conducted with 12-state. Fault detection procedure was conducted with $\chi^2_{\eta,\mathrm{p}} = 23.03$. This value is generated via chi-square distribution with the degree of freedom being 12 and reliability level is 95%. First series of simulation are conducted with noise increment. Figure 4.43 shows EKF with the corrupted measurement. Corruption is simulated by 300 random noise increment with the magnitude of 1E+4.

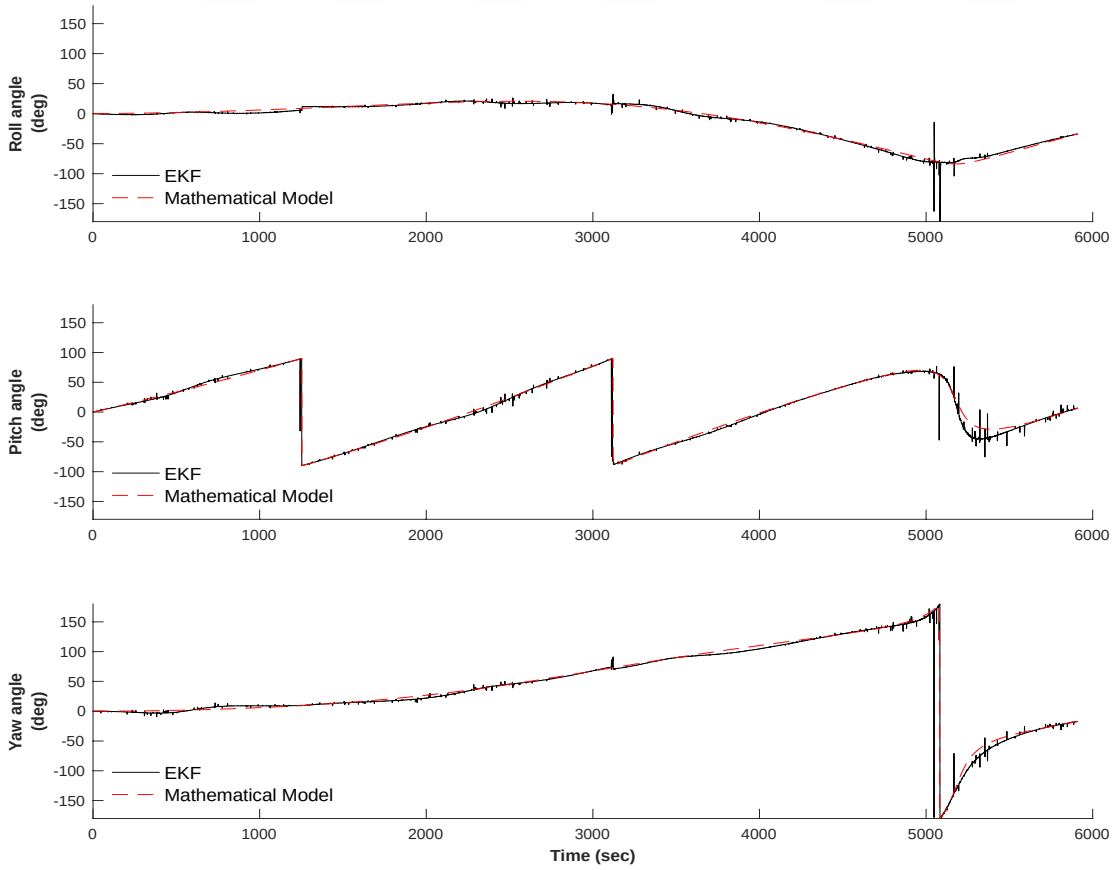

Figure 4.43 : EKF body angle estimations with faulty measurement.

In Figure 4.44-4.45, simulation results after applying adaptive filtering to the EKF and UKF can be seen.

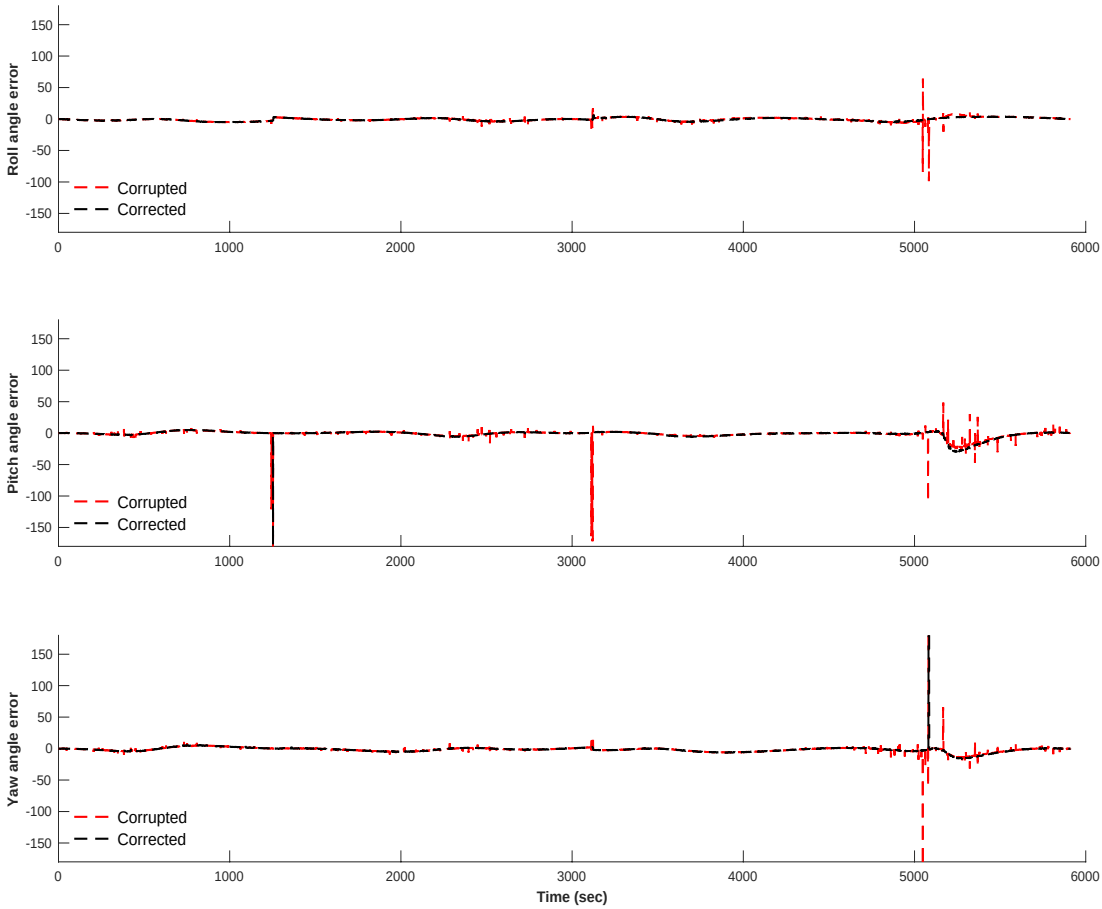

Figure 4.44 : Corrected body estimations with AEKF.

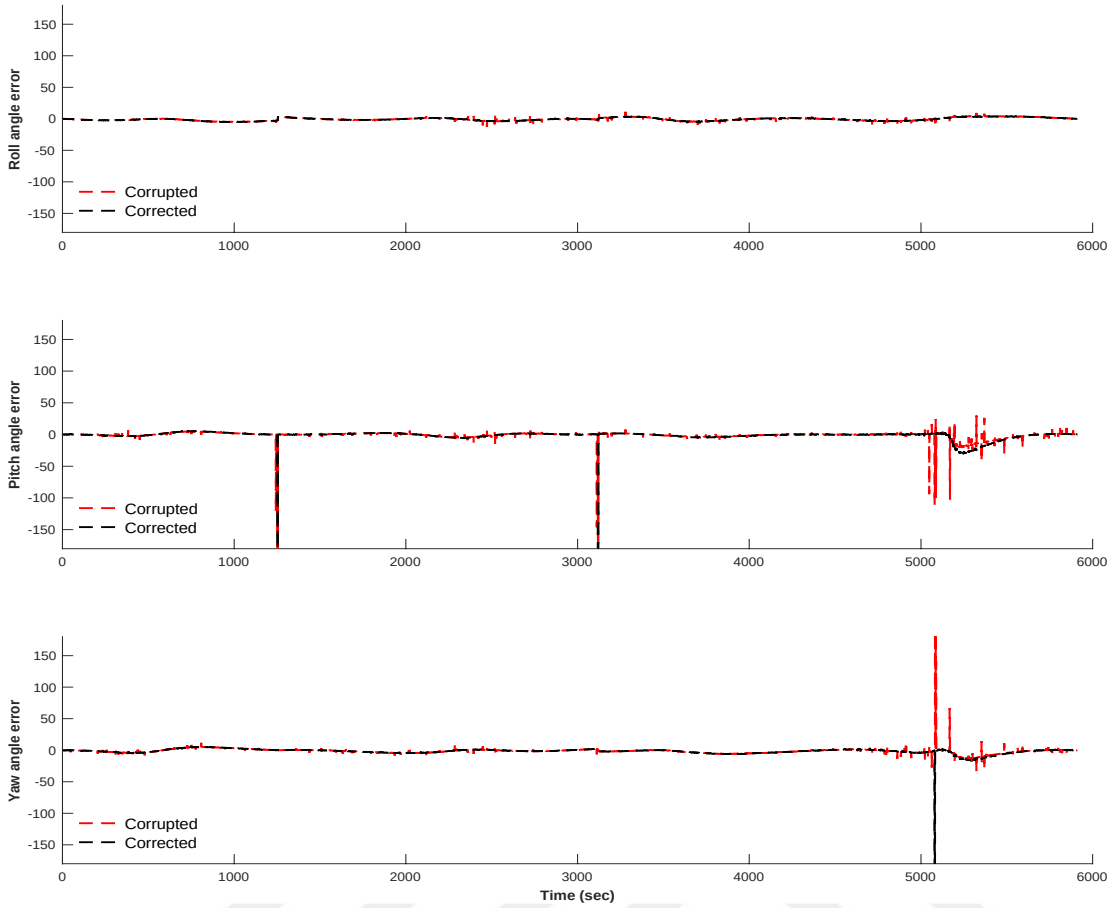

Figure 4.45 : Corrected body estimations with AUKF.

Scaling factor applied to the kalman gain, corrected the faulty measurement effects. Deviations from mathematical model is lower and as can be seen from Figure 4.44-4.45, body angle estimations are corrected dramatically. Same operation can be done to the gyro case. For gyro case, 150 random noise increments with magnitude of 750 are applied to the measurement model. In order to show the effect of adaptive filtering better, error graphs are presented. Figure 4.46-4.47 show body angles graphs.

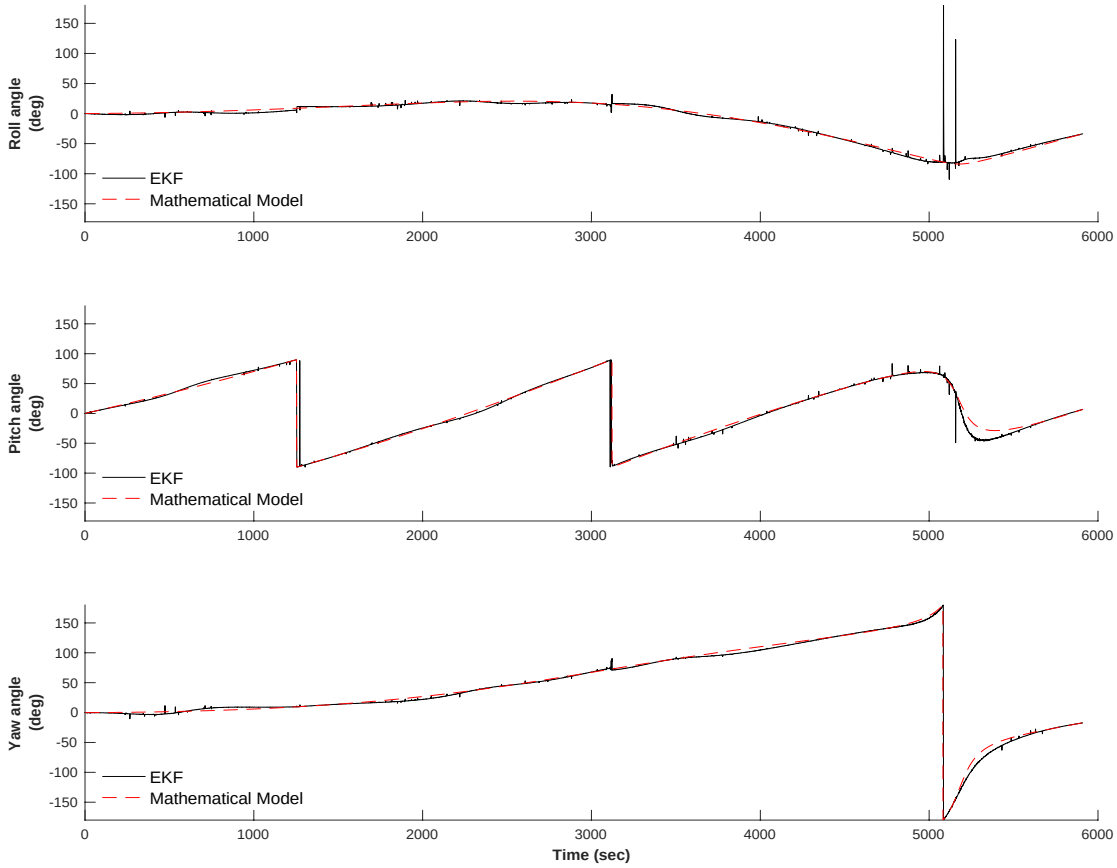

Figure 4.46 : Corrupted EKF body angle estimations.

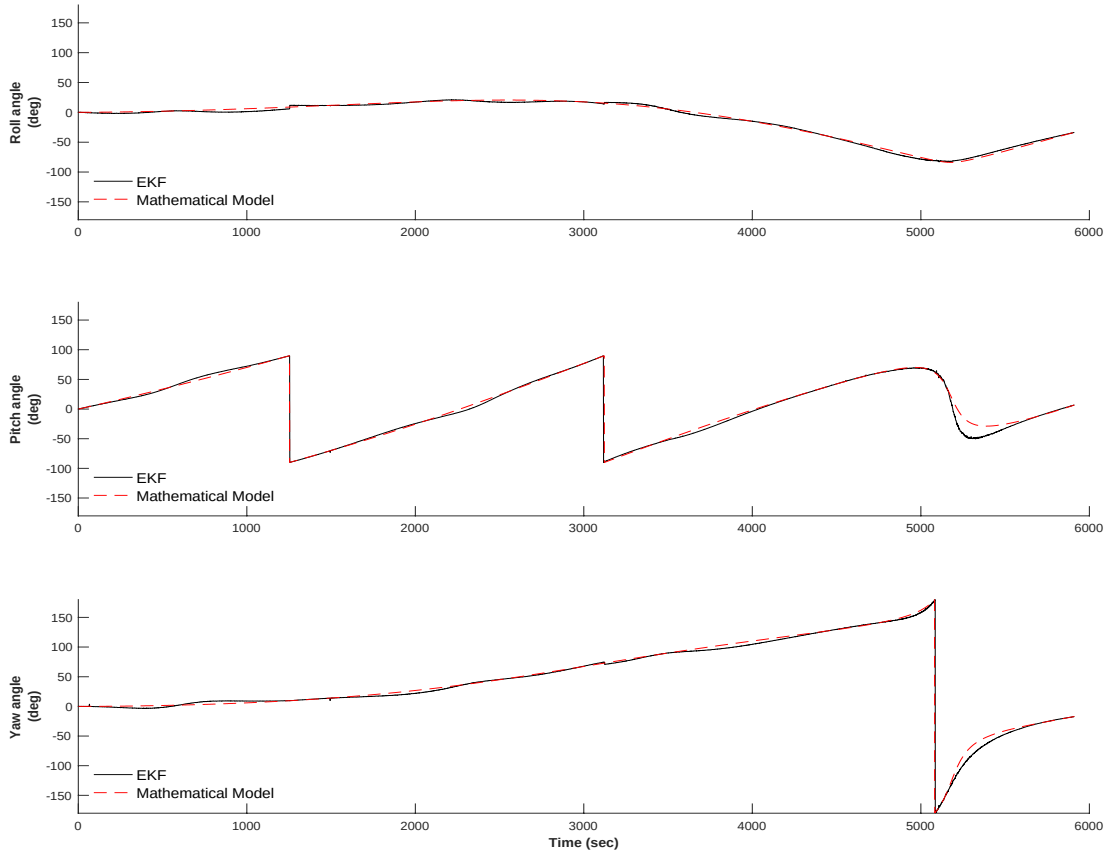

Figure 4.47 : Corrected body angle estimation with AEKF.

Also for this case, adaptive filter eliminated the negative effect of faulty measurements. Positive effect fo the filter can be seen clearly from body angle graph. By 200 random increment points with the magnitude of 150 of the normal noise level, UKF adaptation is presented in Figure 4.48-4.49.

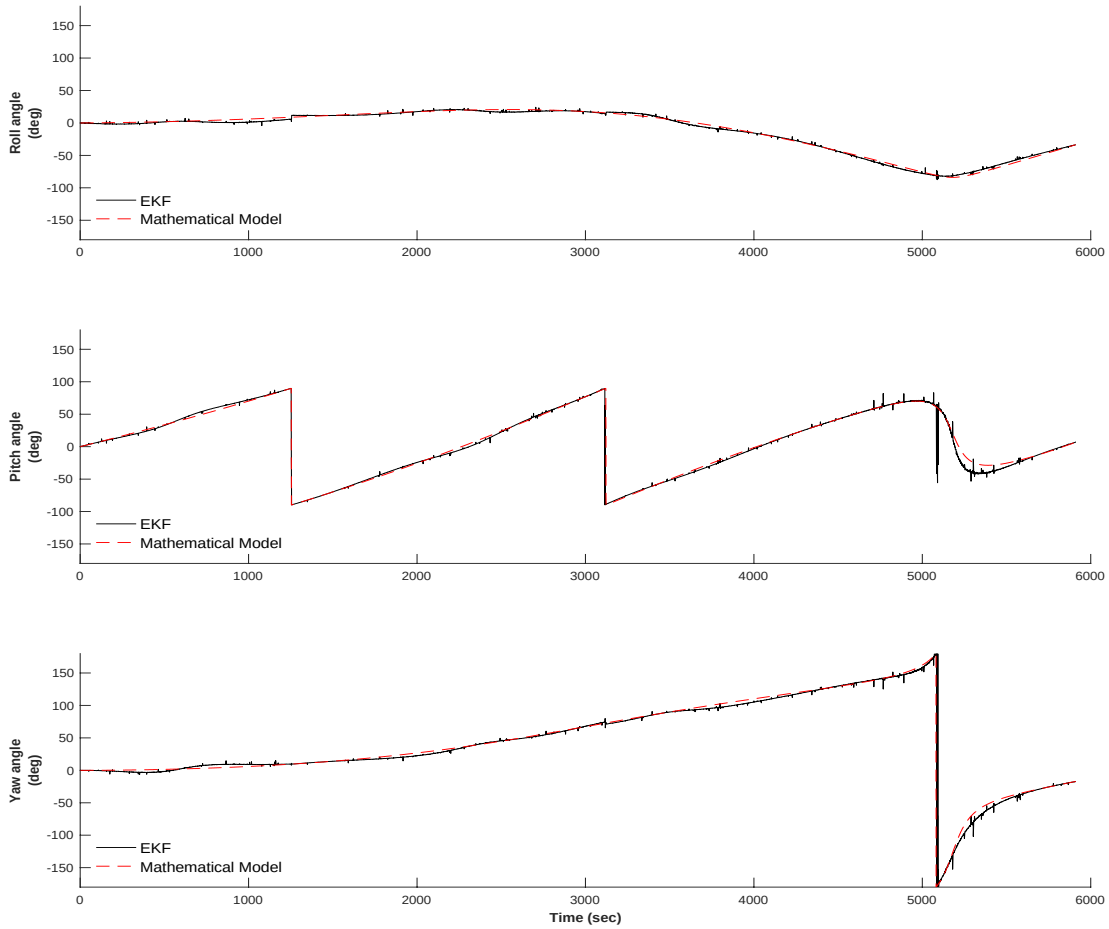

Figure 4.48 : Corrupted UKF body angle estimation.

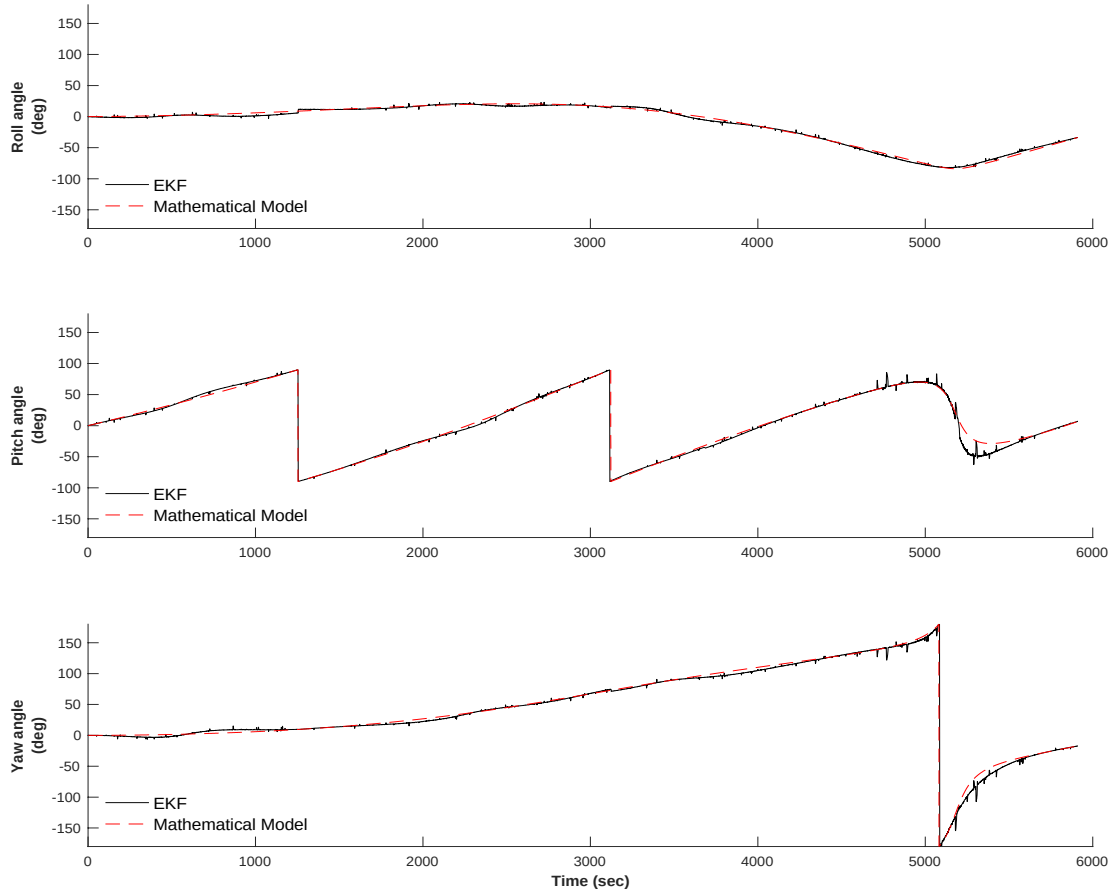

Figure 4.49 : Corrected body angle estimation with AUKF.

For the second set of simulations,bias failure method is used. Using the same random selection process, gyro or magnetometer biases are multipled with different values.By 150 random failure points with magnitude of 100, Figure 4.50-4.51 shows the body angle errors with both UKF and EKF with their respective adaptive versions.

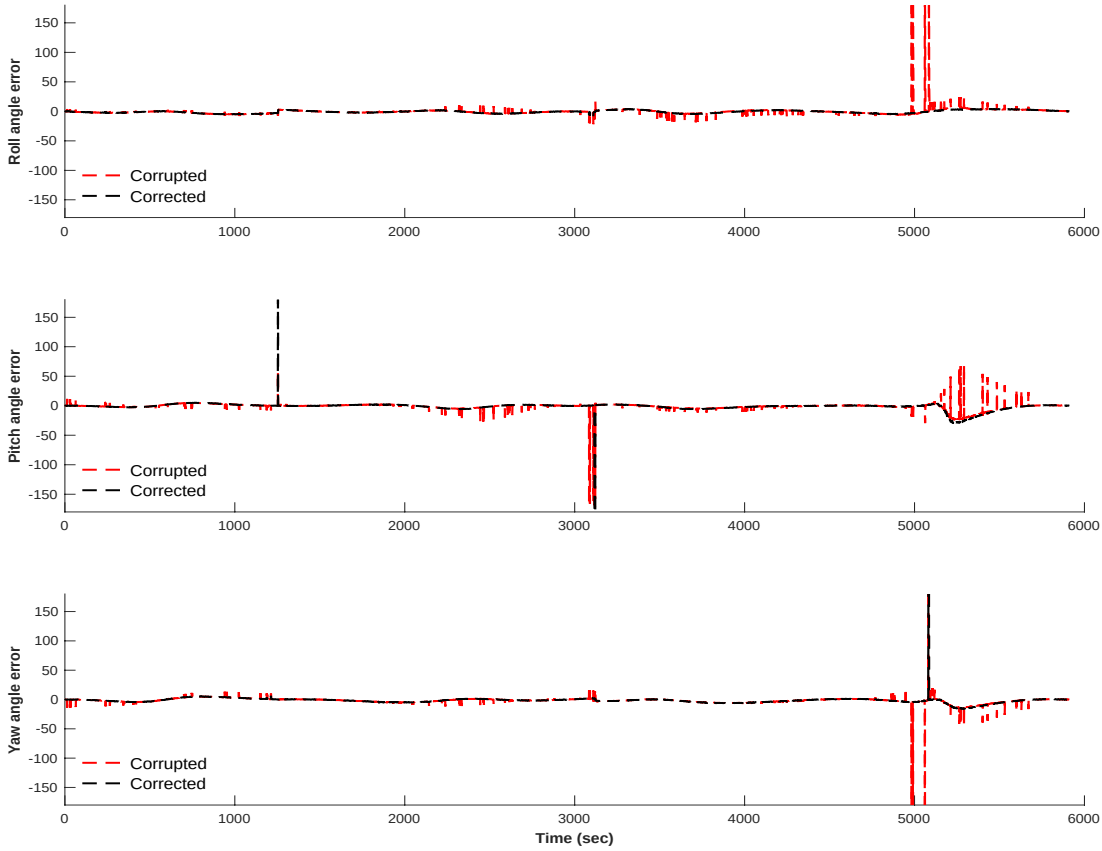

Figure 4.50 : Corrected body estimations with AEKF.

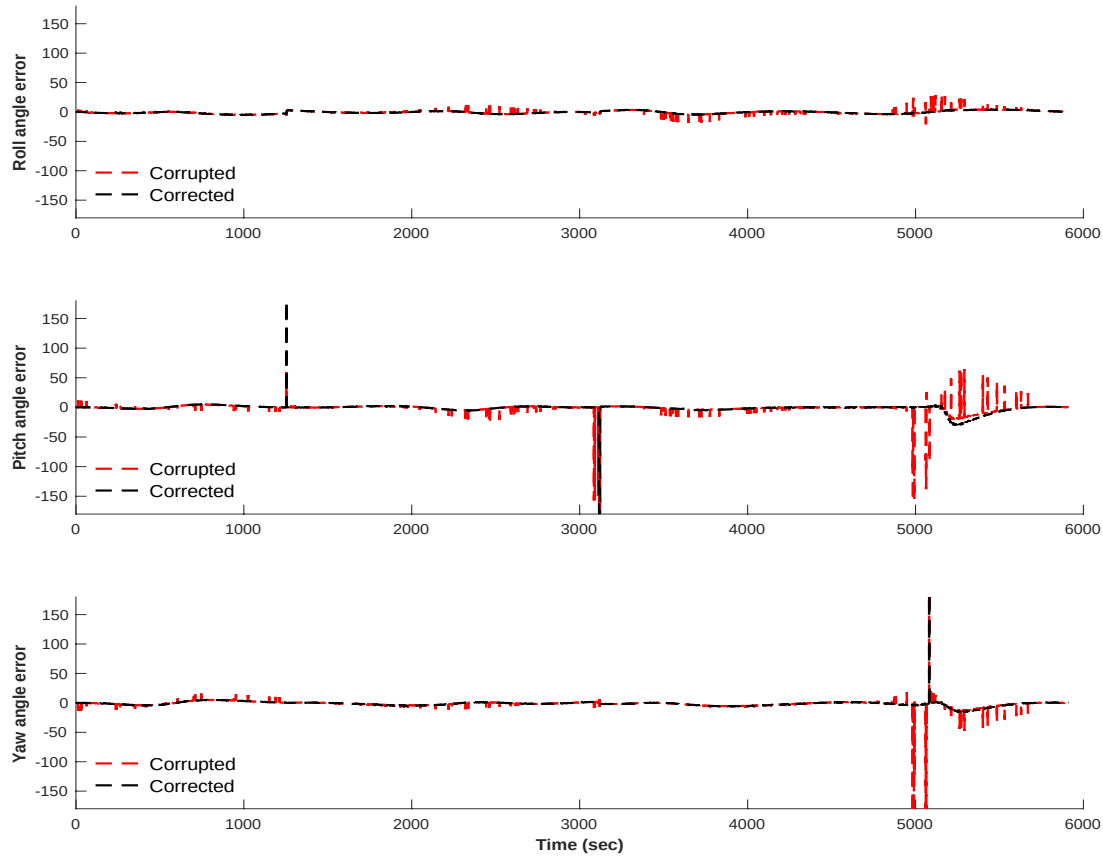

Figure 4.51 : Corrected body estimations with AUKF.

In magnetometer bias failure case, error are much more significant than the noise errors. Algebraic method's effect is considerable. But with the help of adaptive tuning, deviations from mathematical model is close to healthy measurement case. The last graphs in Figure 4.52-4.53. show the body angle error graph with 150 random points of failure by order of 1000.

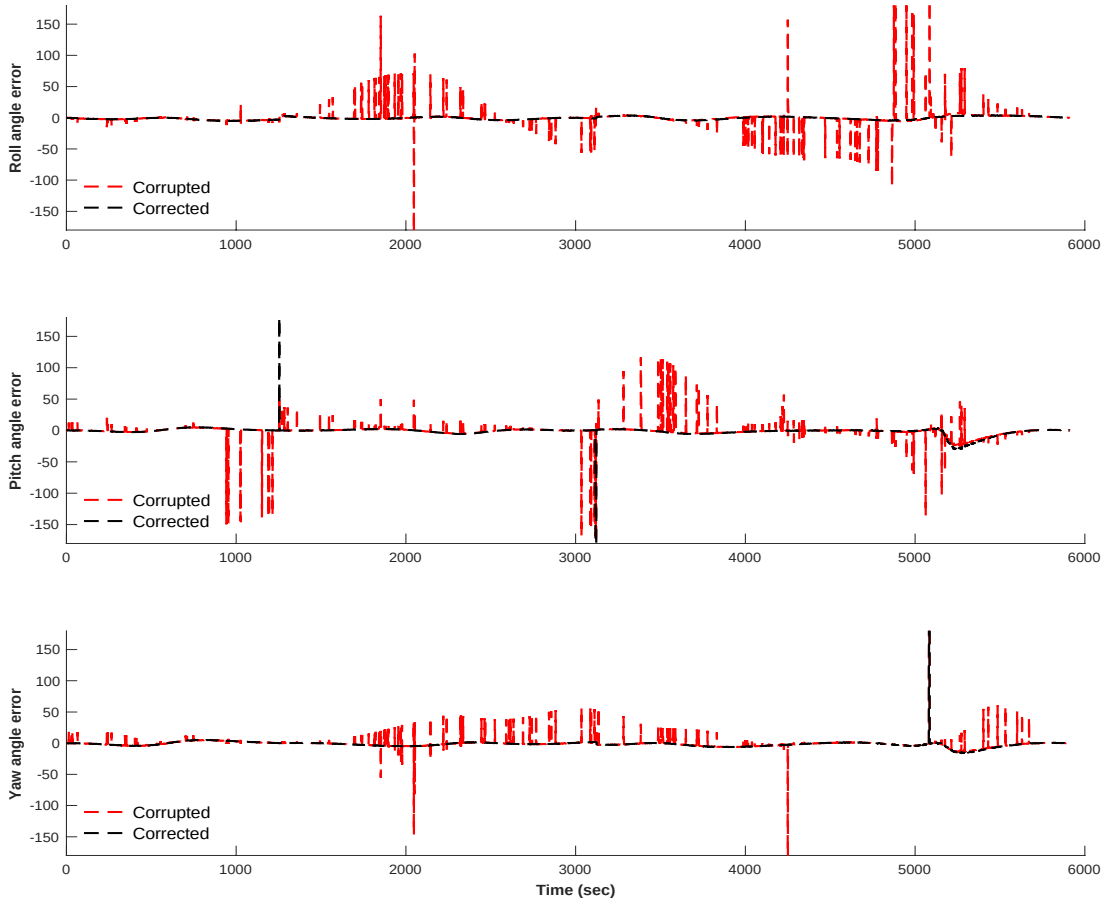

Figure 4.52 : Corrected body estimations with AEKF.

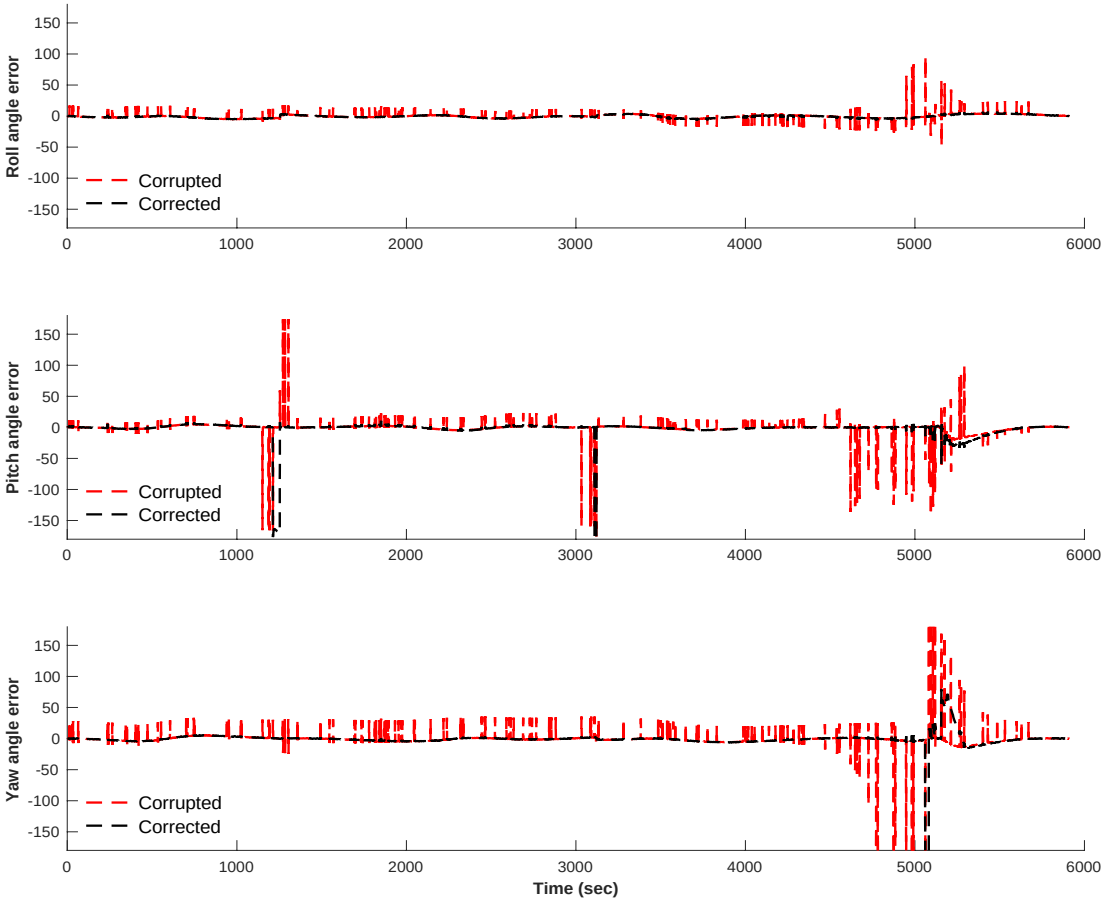

Figure 4.53 : Corrected body estimations with AUKF.

Serious improvements have been observed in both cases. Adaptive methods are decent alternatives to make a robust system but they are not the solution to all of the sensor faults. Too many random point or too much of sensor noise or bias errors caused unstabilities even with the presence of adaptive scale factors. For the last adaptive estimation, instead of using instantenous error pattern, a continuous pattern has been used. For 1000 seconds, error in the magnetometer measurement has been increased by 1E+5 times. Later, adaptive method has been applied to the corrupted measurement. Both filters' estimation performances are given in Figure 4.54-55

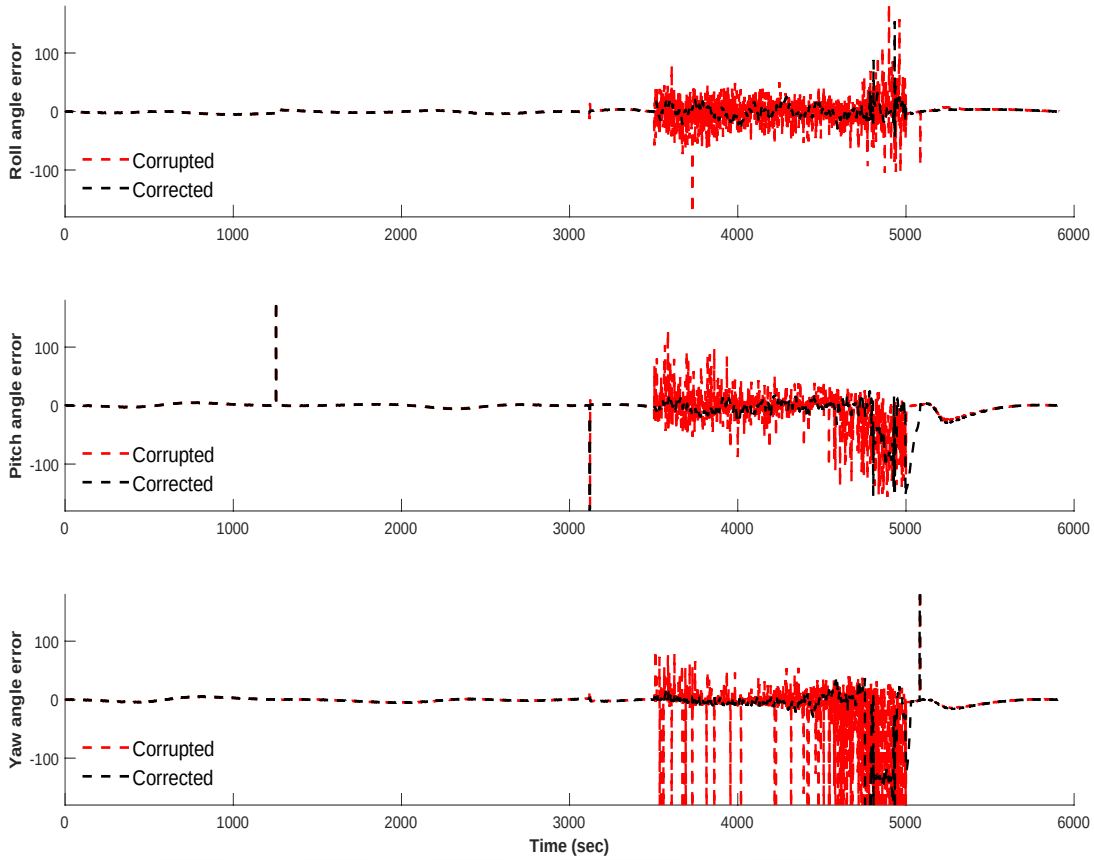

Figure 4.54 : Corrected body estimations with AEKF.

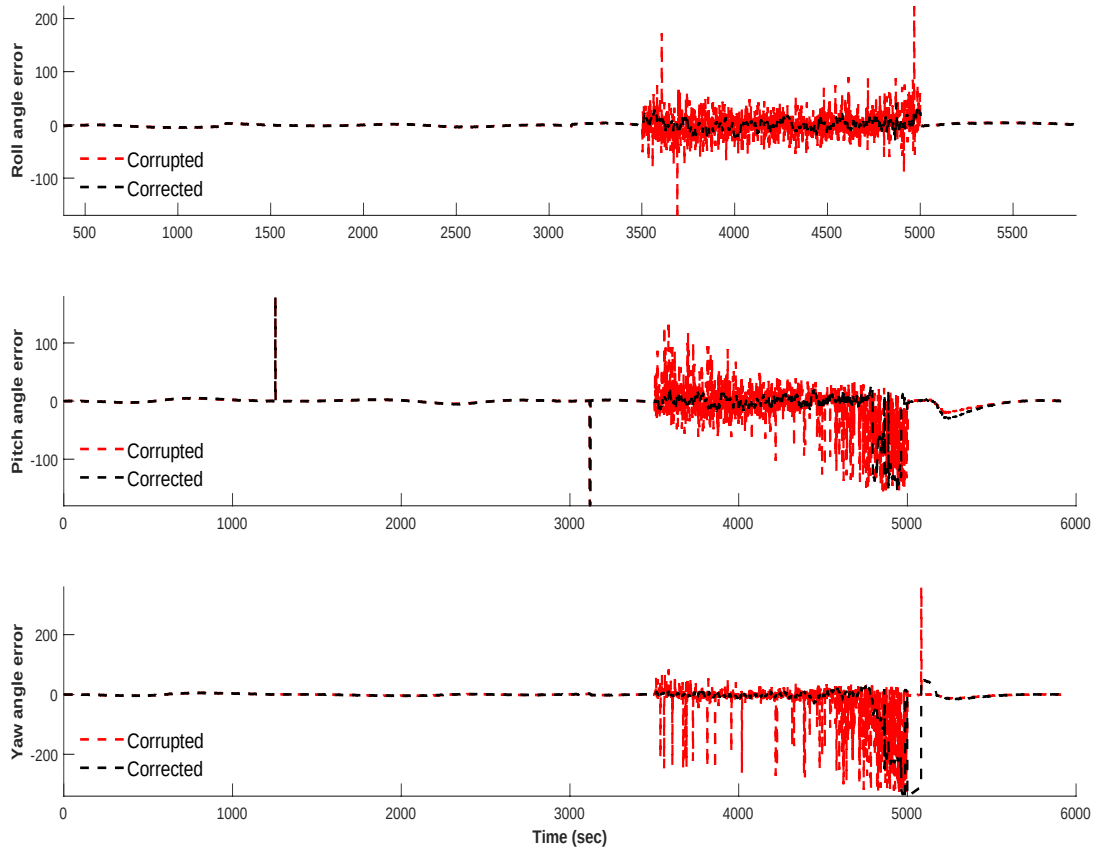

Figure 4.55 : Corrected body estimations with AUKF.

5. CONCLUSIONS

Algebraic method, even though it is an aging algorithm, can estimate satellite attitude well. The sun sensor and magnetometer are selected for inputs to algebraic method because of their wide usage in space industry. Many different filtering algorithms are presented to this day but proven algorithms are still getting attention from engineers. Extended Kalman filter proves itself on many missions. Therefore, selecting the EKF for this work was a must. As expected, it performs really well on simulations especially in low initial angular velocity cases. UKF is known for its superiority to EKF when dealing with nonlinear functions. In many simulations attempts, UKF performed relatively same as EKF. The only significant difference in RMSE values is at roll estimation in 12-state model. Lowering the RMSE values for both case are possible with higher the Q matrix values but this caused oscillations. In this work, better RMSE values are sacrificed for the sake of less ossilations. Another advantage of the using unscented filters is absence of Jacobians. Even though it is fairly small system used in this thesis, calculating jacobians are still open to mistakes. One of the main challenges using UKF was taking the square root of a matrix. It is a costly operation and most of the times, due to round-off errors it fails. Higham's method serves as a decent go around for this problem. The last part of the thesis is adaptive filtering. It is clear that designing a robust system is crucial for long operating times. Even though adaptation methods can be used to estimate certain parameters like inertia matrix of a spacecraft, in this case they are used for measurement and bias correction. If faulty measurement number relatively low than the whole flight time, adaptive filter can correct the errors fairly decent but when faulty sensors produce more data, adaptive methods fail to correct the errros. RMSE tables give a comprehensive picture about which filter is superior to other. In many cases, looking to the RMSE values, UKF performed better. Filters used in this work only estimate Gaussian distribution systems. But for complete probability distribution estimation, nonlinear methods will be invested in the future works.

REFERENCES

Bao, J. & Tsui, Y. (2005). *Fundamentals of Global Positioning System Receivers: A Software Approach.* New York : Wiley.

Black, H.D. (1964). A passive system for determining the attitude of a satellite. *AIAA Journal*. **2**(7), 1350–1351.

Bretagnon, P. and Francou, G. (1988). Planetary Theories in rectangular and spherical variables. VSOP87 solution. Astronomy and Astrophysics. 202,pp.309-315.

Cheng, Y. and Crassidis, J.L. (2004). Particle filtering for sequential spacecraft attitude estimation. *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference*. **4**(August), 2925–2942.

Cilden Guler , D. and Hajiyev, C. (2017). *Gyroless Attitude Determination of Nanosatellites Single-Frame Methods and Kalman Filtering Approach*. Saarbrücken: LAMBERT Academic Publishing.

Cilden-Guler, D., Soken, H. E., Hajiyev, C. (2018). SVD-Aided UKF for Estimation of Nanosatellite Rotational Motion Parameters. *WSEAS Transactions on Signal Processing*. (14), 27-35.

Crassidis, J., Markley, F. and Cheng, Y. (2007). Survey of Nonlinear Attitude Estimation Methods. *Journal of Guidance, Control, and Dynamics*, 30(1), 12-28.

Crassidis, J.L. and Markley, F.L. (2003). Unscented filtering for spacecraft attitude estimation. *AIAA Guidance, Navigation, and Control Conference and Exhibit*. **26**(4).

D'Errico, J. (2020). nearestSPD, MATLAB Central File Exchange. Retrieved May 27, 2020. Available from https://www.mathworks.com/matlabcentral/fileexchange/42885-nearestspd

Doucet, A., Freitas, N.D. and Gordon, N. (2011). Sequential Monte Carlo methods in practice. New York: Springer.

Euler, L. (1775). Formulae generales pro translatione quancunque corporum rigidorum. *Novi commentarii Academiae Scientiarum Imperialis Petropolitanae*. **20**(1776), 189–207.

Farrell, J.L. (1970). Attitude determination by kalman filtering. *Automatica*. **6**(3), 419–430.

H.E. Soken, C.Hacizade (2019). Tuning the Attitude Filter: A Comparison of Intuitive and Adaptive Approaches. *Proc. 9th International Conference on Recent Advances in Space Technologies "Space for the Sustainable Development Goals" (RAST-2019), 11-14 June 2019, Istanbul, Turkey, IEEE*, 747-752.

Hajiyev, C. and Soken, H.E. (2013). Robust Adaptive Kalman Filter for estimation of UAV dynamics in the presence of sensor/actuator faults. *Aerospace Science and Technology*. **28**(1), 376–383.

Hajiyev, C., Soken, H.E. and Cilden-Guler, D. (2019). Nontraditional Attitude Filtering with Simultaneous Process and Measurement Covariance Adaptation. *Journal of Aerospace Engineering*. **32**(5),p.04019054.

Hamilton, W.R. (1866). *Elements of quaternions*. London: Longmans, Green, & co.

Higham, N.J. (1988). Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and Its Applications*. **103**(C), 103–118.

Julier, S.J. (2002). The scaled unscented transformation. *Proceedings of the American Control Conference*. **6**(2), 4555–4559.

Julier, S.J., Uhlmann, J.K. and Durrant-Whyte, H.F. (1995). New approach for filtering nonlinear systems. *Proceedings of the American Control Conference*. **3**(June), 1628–1632.

Kalman, R.E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering, Transactions of the ASME*. **82**(1), 35–45.

Kim, K.H., Lee, J.G. and Park, C.G. (2006). Adaptive two-stage Kalman filter in the presence of unknown random bias. *International Journal of Adaptive Control and Signal Processing*. **20**(7), 305–319.

Kuipers, J.B. (1999). Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality. Princeton, NJ: Princeton University Press.

Lam, Q.M. and Wu, A. (1998). Enhanced precision attitude determination algorithms. *IEEE Aerospace Conference Proceedings*. **1**, 69–76.

Landis, M. (1988). Attitude Determination Using Vector Observations and the Singular Value Decomposition. *Journal of the Astronautical Sciences*. **36**(3), 245–258.

Ma, Z. and Ng, A. (2002). Spacecraft Attitude Determination by Adaptive Kalman Filtering. *AIAA Guidance, Navigation, and Control Conference and Exhibit*. (August).

Markley FL and Crassidis JL (2014). *Fundamentals of spacecraft attitude determination and control.* Springer New York, New York, NY.

Markley, F.L. (2003). Attitude Error Representations for Kalman Filtering. Journal of Guidance, Control, and Dynamics. 26(2),311–317.

Markley, F.L., Berman, N. and Shaked, U. (1994). Deterministic EKF-like estimator for spacecraft attitude estimation. *Proceedings of the American Control Conference*. **1**(0), 247–251.

Mehra, R., Seereeram, S., Bayard, D. and Hadaegh, F. (1995). Adaptive Kalman filtering, failure detection and identification for spacecraft attitude estimation. *IEEE Conference on Control Applications - Proceedings.*, 176–181.

Oshman, Y. and Carmi, A. (2006). Attitude estimation from vector observations using genetic-algorithm- embedded quaternion particle filter. *Journal of Guidance, Control, and Dynamics*. **29**(4), 879–891.

Psiaki, M.L. (2003). Estimation of a spacecraft's attitude dynamics parameters by using flight data. *Journal of Guidance, Control, and Dynamics*. **28**(4), 594–603.

Psiaki, M.L. (2005). Backward-Smoothing Extended Kalman Filter. *Journal of Guidance, Control, and Dynamics*. **28**(5), 885–893.

Rapoport, I. and Oshman, Y. (2002). Optimal filtering in the presence of faulty measurement biases. *Proceedings of the IEEE Conference on Decision and Control*. **2**(December), 2236–2241.

Schmidt, S.F. (1981). The Kalman filter : Its recognition and development for aerospace applications. *Journal of Guidance, Control, and Dynamics*. **4**(1), 4–7.

Sedlak, J. and Chu, D. (1993). Kalman filter estimation of attitude and gyro bias with the QUEST observation model. *Advances in the Astronautical Sciences*. **84**(pt 1), 683–696.

Shuster, M.D. (1993). Survey of attitude representations. *Journal of the Astronautical Sciences*. **41**(4), 439–517.

Shuster, M.D. and Oh, S.D. (1981). Three-axis attitude determination from vector observations. *Journal of Guidance, Control, and Dynamics*. **4**(1), 70–77.

Stuelpnagel, J. (1964). On the Parametrization of the Three-Dimensional Rotation Group. *SIAM Review*. **6**(4), 422–430.

Van Der Merwe, R. and Wan, E.A. (2001). The square-root unscented Kalman filter for state and parameter-estimation. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. **6**, 3461–3464.

Vathsal, S. (1987). Spacecraft attitude determination using a second-order nonlinear filter. *Journal of Guidance, Control, and Dynamics*. **10**(6), 559–566.

Wahba, G. (1965). A Least Squares Estimate of Satellite Attitude. *SIAM Review*. 7(3),409-409.

Wan, E., and van der Merwe, R., (2001) "The Unscented Kalman Filter," Kalman Filtering and Neural Networks, edited by S. Haykin, John Wiley & Sons, New York, NY

Wan, E.A. and Van Der Merwe, R. (2000). The unscented Kalman filter for nonlinear estimation. *IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium, AS-SPCC 2000*., 153–158.

Wertz, J. (1978). *Spacecraft attitude determination and control*. Dordrecht: Kluwer Academic Publishers.

Xia, Q., Rao, M., Ying, Y. and Shen, X. (1994). Adaptive fading Kalman filter with an application. *Automatica*. **30**(8), 1333–1338.

APPENDICES

APPENDIX A: Sensor Measurement Model Source Codes
APPENDIX B: Design Methods Source Codes

APPENDIX A: Sensor Measurement Model Source Codes

A.1. Sun Sensor Measurement Model

```
function sun_orbit = sun(n)
% This function calculates sun direction vector in ECI frame.
    % Period(s)
    N = n*5910;
    time = 1:N;

    %Sun parameters
    us(1,:)=2*pi*1/N:2*pi*1/N:2*pi*N/(1*N);

    % Time
    b = 1;
    t1=datetime(2017,3,16,22,46,22);
    duration = hours(1)+minutes(38)+seconds(29);
    t2=t1+(duration) * n;
    dt=1;

    for periyod=t1:seconds(1):t2

            %Julian Date
        JD(b,1)=367*year(periyod)-
round(7*(year(periyod)+fix((month(periyod)+9)/12))/4)+round(275*mo
nth(periyod)/9)...

+(hour(periyod)+(minute(periyod)/60)+(second(periyod)/3600))/24 +
day(periyod)+1721013.5;

        Tut1(b,1)=(JD(b,1)-2451545)/36525;

        %Mean longitude of the sun
        lambdasun(b,1)=280.460+36000.770*Tut1(b,1);

        Ttdb(b,1)=Tut1(b,1);

        %Mean anomaly of the sun
        Msun(b,1)=357.5277233+35999.05034*Ttdb(b,1);

        %Ecliptic longitude of the sun

lambdaecliptic(b,1)=lambdasun(b,1)+1.914666471*sind(Msun(b,1))+0.0
19994643*sind(2*Msun(b,1));

        %Linear model of the ecliptic of the sun
        ec(b,1)=23.439291-0.0130042*Ttdb(b,1);

        %Unit sun vector
        seci(:,b)=[cosd(lambdaecliptic(b,1));
            sind(lambdaecliptic(b,1))*cosd(ec(b,1));
            sind(lambdaecliptic(b,1))*sind(ec(b,1))];
```

```
        % Transformation matrix (ECI to reference orbit frame)

        A2=[cos(us(1,b))*cos(inc)*cos(lambda)-
sin(us(1,b))*sin(lambda) cos(us(1,b))*sin(inc) -
cos(us(1,b))*cos(inc)*sin(lambda)-sin(us(1,b))*cos(lambda);...
        -sin(inc)*cos(lambda) cos(inc) sin(inc)*sin(lambda);...
        sin(us(1,b))*cos(inc)*cos(lambda)-cos(us(1,b))*sin(lambda)
sin(us(1,b))*sin(inc) -sin(us(1,b))*cos(inc)*sin(lambda)-
cos(us(1,b))*cos(lambda)];

        sun_orbit(:,b)=A2*seci(:,b);

        b=b+1;
    end
```

A.2. Magnetometer Measurement Model

```
function mag_orbit  = magnet(n)
    % Magnetic field model in orbital ref model output

    for t = 1:(5910*n - (n-1))
        %Magneticfield model
        Hx_matrix(t,1) = (M/(r^3))*(cos(w*t)*(cos(e)*sin(inc) -
sin(e)*cos(inc)*cos(we*t)) - sin(w*t)*sin(e)*sin(we*t));
        Hy_matrix(t,1) =  (-
M/(r^3))*(cos(e)*cos(inc)+sin(e)*sin(inc)*cos(we*t));
        Hz_matrix(t,1) =  (2*M/(r^3))*(sin(w*t)*(cos(e)*sin(inc)-
sin(e)*cos(inc)*cos(we*t))-2*sin(w*t)*sin(e)*sin(we*t));

        Hx0_matrix(t,1) =
(1/sqrt(Hx_matrix(t,1)^2+Hy_matrix(t,1)^2+Hz_matrix(t,1)^2))*Hx_ma
trix(t,1);
        Hy0_matrix(t,1) =
(1/sqrt(Hx_matrix(t,1)^2+Hy_matrix(t,1)^2+Hz_matrix(t,1)^2))*Hy_ma
trix(t,1);
        Hz0_matrix(t,1) =
(1/sqrt(Hx_matrix(t,1)^2+Hy_matrix(t,1)^2+Hz_matrix(t,1)^2))*Hz_ma
trix(t,1);
        mag_orbit(:,t) =
[Hx_matrix(t,1);Hy_matrix(t,1);Hz_matrix(t,1)];
    end

end
```

A.3. Gyro Measurement Model

```
      % Angular rates
    wx_matrix(1,periyod+1) = wx_matrix(1,periyod)+
(dt/(Jx))*((wz_matrix(1,periyod)*wy_matrix(1,periyod))*(Jy-
Jz)+NT);
    wy_matrix(1,periyod+1) = wy_matrix(1,periyod)+
(dt/(Jy))*((wx_matrix(1,periyod)*wz_matrix(1,periyod))*(Jz-
Jx)+NT);
    wz_matrix(1,periyod+1) = wz_matrix(1,periyod)+
(dt/(Jz))*((wx_matrix(1,periyod)*wy_matrix(1,periyod))*(Jx-
Jy)+NT);

    % Body Angles
```

```
phi_matrix(1,periyod+1)=phi_matrix(1,periyod)+dt*(wx_matrix(1,peri
yod)*cos(psi_matrix(1,periyod))-
wy_matrix(1,periyod)*sin(psi_matrix(1,periyod)));

theta_matrix(1,periyod+1)=theta_matrix(1,periyod)+dt*(wx_matrix(1,
periyod)*sin(psi_matrix(1,periyod))/cos(phi_matrix(1,periyod))+wy_
matrix(1,periyod)*cos(psi_matrix(1,periyod))/cos(phi_matrix(1,peri
yod))+w);

psi_matrix(1,periyod+1)=psi_matrix(1,periyod)+dt*(tan(phi_matrix(1
,periyod))*(wx_matrix(1,periyod)*sin(psi_matrix(1,periyod))+wy_mat
rix(1,periyod)*cos(psi_matrix(1,periyod)))+wz_matrix(1,periyod));

% Rotation matrix, ref2body
A1 = [ cos(psi_matrix(1,periyod))*cos(theta_matrix(1,periyod)) +
sin(phi_matrix(1,periyod))*sin(psi_matrix(1,periyod))*sin(theta_ma
trix(1,periyod)),
cos(phi_matrix(1,periyod))*sin(psi_matrix(1,periyod)),
cos(theta_matrix(1,periyod))*sin(phi_matrix(1,periyod))*sin(psi_ma
trix(1,periyod)) -
cos(psi_matrix(1,periyod))*sin(theta_matrix(1,periyod))
cos(psi_matrix(1,periyod))*sin(phi_matrix(1,periyod))*sin(theta_ma
trix(1,periyod)) -
cos(theta_matrix(1,periyod))*sin(psi_matrix(1,periyod)),
cos(phi_matrix(1,periyod))*cos(psi_matrix(1,periyod)),
sin(psi_matrix(1,periyod))*sin(theta_matrix(1,periyod)) +
cos(psi_matrix(1,periyod))*cos(theta_matrix(1,periyod))*sin(phi_ma
trix(1,periyod))

cos(phi_matrix(1,periyod))*sin(theta_matrix(1,periyod)),
-sin(phi_matrix(1,periyod)),
cos(phi_matrix(1,periyod))*cos(theta_matrix(1,periyod))];

        rates(:,periyod) =
[wx_matrix(1,periyod);wy_matrix(1,periyod);wz_matrix(1,periyod)];
        sigmam = 0.0003;
        error_gyro = sigmam*randn(3,1);
        gyro_body(:,periyod) = rates(:,periyod) + error_gyro +
bg(:,1);
```

APPENDIX B: Design Methods Source Codes

B.1. Algebraic Method

```
    % Orbital ref. triad
    t1o = sun_orbit(:,periyod);
    t2o = cross(sun_orbit(:,periyod),
mag_orbit(:,periyod))/norm(cross(sun_orbit(:,periyod),
mag_orbit(:,periyod)));
    t3o = cross(t1o,t2o);

    % Body ref. triad
    t1b = sun_body(:,periyod);
    t2b = cross(sun_body(:,periyod),
mag_body(:,periyod))/norm(cross(sun_body(:,periyod),
mag_body(:,periyod)));
    t3b = cross(t1b,t2b);

    % Attitutde Matrix
    A(:,:,periyod) = [t1b, t2b, t3b]*[t1o, t2o, t3o]';

%     roll(periyod) = real(asin(-A(3,2,periyod)));
    yaw(periyod) = atan2(A(1,2,periyod),A(2,2,periyod));
    pitch(periyod) = atan(A(3,1,periyod)/A(3,3,periyod));
    roll(periyod) = atan(-
A(3,2,periyod)*cos(yaw(periyod))/A(2,2,periyod));

    %Covariance
    K(:,:,periyod) = (sigmac^2 *
sun_body(:,periyod)*sun_body(:,periyod)' + sigmas^2 *
mag_body(:,periyod) *
mag_body(:,periyod)')/(norm(cross(sun_body(:,periyod),
mag_body(:,periyod)))^2) ...
    + sigmas^2 * t2b * t2b' ;
```

B.2. Extended Kalman filter

```
   % Update apriori estimate
    % Angular Rates
    x_apriori(4,i) = x_aposteriori(4,i-1)+
(dt/(Jx))*((x_aposteriori(6,i-1)*x_aposteriori(5,i-1))*(Jy-Jz) +
NT);
    x_apriori(5,i) = x_aposteriori(5,i-1)+
(dt/(Jy))*((x_aposteriori(4,i-1)*x_aposteriori(6,i-1))*(Jz-Jx) +
NT);
    x_apriori(6,i) = x_aposteriori(6,i-1)+
(dt/(Jz))*((x_aposteriori(4,i-1)*x_aposteriori(5,i-1))*(Jx-Jy) +
NT);

    % Body Angles
    x_apriori(1,i) = x_aposteriori(1,i-1)+dt*(x_aposteriori(4,i-
1)*cos(x_aposteriori(3,i-1))-x_aposteriori(5,i-
1)*sin(x_aposteriori(3,i-1)));
```

```
    x_apriori(2,i) = x_aposteriori(2,i-1)+dt*(x_aposteriori(4,i-
1)*sin(x_aposteriori(3,i-1))/cos(x_aposteriori(1,i-
1))+x_aposteriori(5,i-1)*cos(x_aposteriori(3,i-
1))/cos(x_aposteriori(1,i-1))+w);
    x_apriori(3,i) = x_aposteriori(3,i-
1)+dt*(tan(x_aposteriori(1,i-1))*(x_aposteriori(4,i-
1)*sin(x_aposteriori(3,i-1))+x_aposteriori(5,i-
1)*cos(x_aposteriori(3,i-1)))+x_aposteriori(6,i-1));

    % Biases
    % Magnetometer
    x_apriori(7,i) = x_aposteriori(7,i-1);
    x_apriori(8,i) = x_aposteriori(8,i-1);
    x_apriori(9,i) = x_aposteriori(9,i-1);

    % Gyro
    x_apriori(10,i) = x_aposteriori(10,i-1);
    x_apriori(11,i) = x_aposteriori(11,i-1);
    x_apriori(12,i) = x_aposteriori(12,i-1);
    %-------------------------------------------------------------
    % Update state Jacobian
    F = jacobian(x_aposteriori(1,i-1),x_aposteriori(2,i-
1),x_aposteriori(3,i-1),x_aposteriori(4,i-1),x_aposteriori(5,i-
1),x_aposteriori(6,i-1),dt);
    %-------------------------------------------------------------
    % Update aprioiri error covariance estimate
    P_apriori(:,:,i) = F*P_aposteriori(:,:,i-1)*F' + Q;
    %-------------------------------------------------------------
    % Measurement-state Jacobian
    H = h_jacobian(x_apriori(1,i),x_apriori(2,i),x_apriori(3,i),
mag_orbit(1,i),mag_orbit(2,i),mag_orbit(3,i));
    %-------------------------------------------------------------
    A3 = [ cos(x_apriori(3,i))*cos(x_apriori(2,i)) +
sin(x_apriori(1,i))*sin(x_apriori(3,i))*sin(x_apriori(2,i)),
cos(x_apriori(1,i))*sin(x_apriori(3,i)),
cos(x_apriori(2,i))*sin(x_apriori(1,i))*sin(x_apriori(3,i)) -
cos(x_apriori(3,i))*sin(x_apriori(2,i))

cos(x_apriori(3,i))*sin(x_apriori(1,i))*sin(x_apriori(2,i)) -
cos(x_apriori(2,i))*sin(x_apriori(3,i)),
cos(x_apriori(1,i))*cos(x_apriori(3,i)),
sin(x_apriori(3,i))*sin(x_apriori(2,i)) +
cos(x_apriori(3,i))*cos(x_apriori(2,i))*sin(x_apriori(1,i))
        cos(x_apriori(1,i))*sin(x_apriori(2,i)),-
sin(x_apriori(1,i)),cos(x_apriori(1,i))*cos(x_apriori(2,i))];


    % Update Kalman gain
    Kg(:,:,i) = P_apriori(:,:,i)*H' /
(H*P_apriori(:,:,i)*H'+R(:,:,i));

    z_model(:,i) = [eye(3)*x_apriori(1:3,i);
                    (eye(3)*x_apriori(4:6,i) + x_apriori(10:12,i)
- x_aposteriori(10:12,i-1));
                    (A3*mag_orbit(:,i) + x_apriori(7:9,i)  -
x_aposteriori(7:9, i-1))];

    inno(:,i) = z(:,i) - z_model(:,i);
```

```
    % Update aposteriori state estimate
    x_aposteriori(:,i) = x_apriori(:,i) + Kg(:,:,i) * (inno(:,i));

    %-------------------------------------------------------------
-------------
    % Update aposteriori error covariance estimate
    P_aposteriori(:,:,i) = (eye(length(x_apriori(:,i))) -
Kg(:,:,i)*H) * P_apriori(:,:,i);
```

B.3. Unscented Kalman filter

```
    % Sigma points calculations
    Wm = [delta/(L+delta) 0.5/(L+delta)+zeros(1,2*L)];
    Wc = Wm;
    Wc(1) = Wc(1) + (1-alfa^2+beta);

    % sigma points
     [R_chol,p_chol] = chol(P_aposteriori(:,:,i-1)+ Q );
    if p_chol  == 0
    T=sqrt((L+delta))*chol(P_aposteriori(:,:,i-1) + Q );
    else
    T=sqrt((L+delta))*chol(nearestSPD(P_aposteriori(:,:,i-1)+ Q
));
    end
    sgp = [x_aposteriori(:,i-1) x_aposteriori(:,i-1)+T
x_aposteriori(:,i-1)-T];

    phi_sg = sgp(1,:);
    theta_sg = sgp(2,:);
    psi_sg = sgp(3,:);
    wx_sg= sgp(4,:);
    wy_sg= sgp(5,:);
    wz_sg= sgp(6,:);
    bcx_sg = sgp(7,:);
    bcy_sg = sgp(8,:);
    bcz_sg = sgp(9,:);
    bgx_sg = sgp(10,:);
    bgy_sg = sgp(11,:);
    bgz_sg = sgp(12,:);

    %-------------------------------------------------------------
    % Sigma point propagation to next time step.
    for k = 1:size(sgp,2)
        % Angular rates
        sig_apriori(4,k) = wx_sg(k)+
(dt/(Jx))*((wz_sg(k)*wy_sg(k))*(Jy-Jz)+NT);
        sig_apriori(5,k) = wy_sg(k)+
(dt/(Jy))*((wx_sg(k)*wz_sg(k))*(Jz-Jx)+NT);
        sig_apriori(6,k) = wz_sg(k)+
(dt/(Jz))*((wx_sg(k)*wy_sg(k))*(Jx-Jy)+NT);

        % Body Angles
        sig_apriori(1,k)=phi_sg(k)+dt*(wx_sg(k)*cos(psi_sg(k))-
wy_sg(k)*sin(psi_sg(k)));

sig_apriori(2,k)=theta_sg(k)+dt*(wx_sg(k)*sin(psi_sg(k))/cos(phi_s
g(k))+wy_sg(k)*cos(psi_sg(k))/cos(phi_sg(k))+w);
```

```
sig_apriori(3,k)=psi_sg(k)+dt*(tan(phi_sg(k))*(wx_sg(k)*sin(psi_sg
(k))+wy_sg(k)*cos(psi_sg(k)))+wz_sg(k));

        sig_apriori(1,k) = wrapToPi(sig_apriori(1,k));
        sig_apriori(3,k) = wrapToPi(sig_apriori(3,k));


        % Magnetometer Bias
        sigma = 1e-4;
        sig_apriori(7,k) = bcx_sg(k); %+ sigma*randn(1,1);
        sig_apriori(8,k) = bcy_sg(k); %+ sigma*randn(1,1);
        sig_apriori(9,k) = bcz_sg(k); %+ sigma*randn(1,1);

        % Gyro Bias
        sig_apriori(10,k) = bgx_sg(k);
        sig_apriori(11,k) = bgy_sg(k);
        sig_apriori(12,k) = bgz_sg(k);
    end

    %forecast state
    x_apriori = 0;
    x_apriori = Wm(1)*sig_apriori(:,1);
    for k = 2:size(sig_apriori,2)
        x_apriori = x_apriori+Wm(k)*sig_apriori(:,k);
    end

    %-------------------------------------------------------------
    % Covariance prior estimation
    P_apriori(:,:) =
zeros(size(sig_apriori,1),size(sig_apriori,1));
    P_apriori(:,:) = Wc(1)*(sig_apriori(:,1) -
x_apriori)*(sig_apriori(:,1) - x_apriori)';
    for k = 2:size(sig_apriori,2)
        P_apriori(:,:) = P_apriori(:,:) + Wc(k)*((sig_apriori(:,k)
- x_apriori)*(sig_apriori(:,k) - x_apriori)');
    end

    %-------------------------------------------------------------
    % Measurement sigma point
    meas_sig_points = 0;
    for k = 1:size(sig_apriori,2)
  A3 = [ cos(sig_apriori(3,k))*cos(sig_apriori(2,k)) +
sin(sig_apriori(1,k))*sin(sig_apriori(3,k))*sin(sig_apriori(2,k)),
cos(sig_apriori(1,k))*sin(sig_apriori(3,k)),
cos(sig_apriori(2,k))*sin(sig_apriori(1,k))*sin(sig_apriori(3,k))
- cos(sig_apriori(3,k))*sin(sig_apriori(2,k))

cos(sig_apriori(3,k))*sin(sig_apriori(1,k))*sin(sig_apriori(2,k))
- cos(sig_apriori(2,k))*sin(sig_apriori(3,k)),
cos(sig_apriori(1,k))*cos(sig_apriori(3,k)),
sin(sig_apriori(3,k))*sin(sig_apriori(2,k)) +
cos(sig_apriori(3,k))*cos(sig_apriori(2,k))*sin(sig_apriori(1,k))
            cos(sig_apriori(1,k))*sin(sig_apriori(2,k)),-
sin(sig_apriori(1,k)),cos(sig_apriori(1,k))*cos(sig_apriori(2,k))]
;

        % Angle and Rate Measurements
        meas_sig_points(1:3,k) = sig_apriori(1:3,k);
```

```
   meas_sig_points(4:6,k) =
sig_apriori(4:6,k)+sig_apriori(10:12,k) - x_aposteriori(10:12,i-
1);
    meas_sig_points(7:9,k) = A3*mag_orbit(:,i) +sig_apriori(7:9,k)
- x_aposteriori(7:9,i-1);

    end

    %-------------------------------------------------------------
    % Measurement sigma points estimation
    z_sig = 0;
    z_sig = Wm(1)*meas_sig_points(:,1);
    for k = 2:size(sig_apriori,2)
        z_sig = z_sig + Wm(k)*meas_sig_points(:,k);
    end

    %-------------------------------------------------------------
    % Variance and Covariance calculation
    % Measurement variance

    Pzz = 0;
    Pzz =  Wc(1)*((meas_sig_points(:,1) -
z_sig)*(meas_sig_points(:,1) - z_sig)');
    for k = 2:size(sig_apriori,2)
        Pzz = Pzz + Wc(k)*((meas_sig_points(:,k) -
z_sig)*(meas_sig_points(:,k) - z_sig)');
    end
    Pz = Pzz + R(:,:,i);

    % State-Measurement covariance

    Pxz = zeros(size(sig_apriori,1),size(z_sig,1),1);
    Pxz =  Wc(1)*((sig_apriori(:,1) -
x_apriori)*(meas_sig_points(:,1) - z_sig)');
    for k = 2:size(sig_apriori,2)
        Pxz = Pxz + Wc(k)*((sig_apriori(:,k) -
x_apriori)*(meas_sig_points(:,k) - z_sig)');
    end

    %-------------------------------------------------------------
    % Kalman Gain
    Kg(:,:,i) = Pxz/Pz;

    inno = [z(1:3,i) - z_sig(1:3);
            z(4:6,i) - z_sig(4:6) ;
            z(7:9,i) - z_sig(7:9) ];

    %-------------------------------------------------------------
    % Update Apriori state
    x_aposteriori(:,i) = x_apriori + Kg(:,:,i)*inno;

    %-------------------------------------------------------------
    % Update Apriori covariance

    P_aposteriori(:,:,i) = P_apriori(:,:) -
Kg(:,:,i)*Pz*Kg(:,:,i)';
```

B.4. Adaptive Extended Kalman filter

```
    % Update apriori estimate
     % Angular Rates
     x_apriori(4,i) = x_aposteriori(4,i-1)+
(dt/(Jx))*((x_aposteriori(6,i-1)*x_aposteriori(5,i-1))*(Jy-Jz) +
NT);
     x_apriori(5,i) = x_aposteriori(5,i-1)+
(dt/(Jy))*((x_aposteriori(4,i-1)*x_aposteriori(6,i-1))*(Jz-Jx) +
NT);
     x_apriori(6,i) = x_aposteriori(6,i-1)+
(dt/(Jz))*((x_aposteriori(4,i-1)*x_aposteriori(5,i-1))*(Jx-Jy) +
NT);

     % Body Angles
     x_apriori(1,i) = x_aposteriori(1,i-1)+dt*(x_aposteriori(4,i-
1)*cos(x_aposteriori(3,i-1))-x_aposteriori(5,i-
1)*sin(x_aposteriori(3,i-1)));
     x_apriori(2,i) = x_aposteriori(2,i-1)+dt*(x_aposteriori(4,i-
1)*sin(x_aposteriori(3,i-1))/cos(x_aposteriori(1,i-
1))+x_aposteriori(5,i-1)*cos(x_aposteriori(3,i-
1))/cos(x_aposteriori(1,i-1))+w);
     x_apriori(3,i) = x_aposteriori(3,i-
1)+dt*(tan(x_aposteriori(1,i-1))*(x_aposteriori(4,i-
1)*sin(x_aposteriori(3,i-1))+x_aposteriori(5,i-
1)*cos(x_aposteriori(3,i-1)))+x_aposteriori(6,i-1));

     % Biases
     % Magnetometer
     x_apriori(7,i) = x_aposteriori(7,i-1);
     x_apriori(8,i) = x_aposteriori(8,i-1);
     x_apriori(9,i) = x_aposteriori(9,i-1);

     % Gyro
     x_apriori(10,i) = x_aposteriori(10,i-1);
     x_apriori(11,i) = x_aposteriori(11,i-1);
     x_apriori(12,i) = x_aposteriori(12,i-1);
     %-------------------------------------------------------------
     % Update state Jacobian
     F = jacobian(x_aposteriori(1,i-1),x_aposteriori(2,i-
1),x_aposteriori(3,i-1),x_aposteriori(4,i-1),x_aposteriori(5,i-
1),x_aposteriori(6,i-1),dt);
     %-------------------------------------------------------------
     % Update aprioiri error covariance estimate
     P_apriori(:,:,i) = F*P_aposteriori(:,:,i-1)*F' + Q;
     %-------------------------------------------------------------
     % Measurement-state Jacobian
     H = h_jacobian(x_apriori(1,i),x_apriori(2,i),x_apriori(3,i),
mag_orbit(1,i),mag_orbit(2,i),mag_orbit(3,i));
     %-------------------------------------------------------------
     A3 = [ cos(x_apriori(3,i))*cos(x_apriori(2,i)) +
sin(x_apriori(1,i))*sin(x_apriori(3,i))*sin(x_apriori(2,i)),
cos(x_apriori(1,i))*sin(x_apriori(3,i)),
cos(x_apriori(2,i))*sin(x_apriori(1,i))*sin(x_apriori(3,i)) -
cos(x_apriori(3,i))*sin(x_apriori(2,i))

cos(x_apriori(3,i))*sin(x_apriori(1,i))*sin(x_apriori(2,i)) -
cos(x_apriori(2,i))*sin(x_apriori(3,i)),
cos(x_apriori(1,i))*cos(x_apriori(3,i)),
sin(x_apriori(3,i))*sin(x_apriori(2,i)) +
cos(x_apriori(3,i))*cos(x_apriori(2,i))*sin(x_apriori(1,i))
```

```
        cos(x_apriori(1,i))*sin(x_apriori(2,i)),-
sin(x_apriori(1,i)),cos(x_apriori(1,i))*cos(x_apriori(2,i))];


    % Update Kalman gain
    Kg(:,:,i) = P_apriori(:,:,i)*H' /
(H*P_apriori(:,:,i)*H'+R(:,:,i));

    z_model(:,i) = [eye(3)*x_apriori(1:3,i);
                    (eye(3)*x_apriori(4:6,i) + x_apriori(10:12,i)
- x_aposteriori(10:12,i-1));
                    (A3*mag_orbit(:,i) + x_apriori(7:9,i)  -
x_aposteriori(7:9, i-1))];

    inno(:,i) = z(:,i) - z_model(:,i);

    ind(i) = inno' * 1/(H*P_apriori(:,:,i)*H'+R(:,:,i)) * inno;


        if ind(i) > 23.03
           Ck = H*P_apriori(:,:,i)*H';
           Ck_hat = inno*inno';
           Sk(:,i) = max(1,diag(Ck_hat)./diag(Ck));
               for v = 1:12
                   for j = 1:9
                           Kg(v,j,i) = Kg(v,j,i)/Sk(j,i);
                   end
               end
        end

    % Update aposteriori state estimate
    x_aposteriori(:,i) = x_apriori(:,i) + Kg(:,:,i) * (inno(:,i));

    %-----------------------------------------------------------------
    % Update aposteriori error covariance estimate
    P_aposteriori(:,:,i) = (eye(length(x_apriori(:,i))) -
Kg(:,:,i)*H) * P_apriori(:,:,i);
```

B.5. Adaptive Unscented Kalman filter

```
    % Sigma points calculations
    Wm = [delta/(L+delta) 0.5/(L+delta)+zeros(1,2*L)];
    Wc = Wm;
    Wc(1) = Wc(1) + (1-alfa^2+beta);

    % sigma points
     [R_chol,p_chol] = chol(P_aposteriori(:,:,i-1)+ Q );
    if p_chol  == 0
    T=sqrt((L+delta))*chol(P_aposteriori(:,:,i-1) + Q );
    else
    T=sqrt((L+delta))*chol(nearestSPD(P_aposteriori(:,:,i-1)+ Q
));
    end
    sgp = [x_aposteriori(:,i-1) x_aposteriori(:,i-1)+T
x_aposteriori(:,i-1)-T];
```

```
    phi_sg = sgp(1,:);
    theta_sg = sgp(2,:);
    psi_sg = sgp(3,:);
    wx_sg= sgp(4,:);
    wy_sg= sgp(5,:);
    wz_sg= sgp(6,:);
    bcx_sg = sgp(7,:);
    bcy_sg = sgp(8,:);
    bcz_sg = sgp(9,:);
    bgx_sg = sgp(10,:);
    bgy_sg = sgp(11,:);
    bgz_sg = sgp(12,:);

    %-------------------------------------------------------------
    % Sigma point propagation to next time step.
    for k = 1:size(sgp,2)
        % Angular rates
        sig_apriori(4,k) = wx_sg(k)+
(dt/(Jx))*((wz_sg(k)*wy_sg(k))*(Jy-Jz)+NT);
        sig_apriori(5,k) = wy_sg(k)+
(dt/(Jy))*((wx_sg(k)*wz_sg(k))*(Jz-Jx)+NT);
        sig_apriori(6,k) = wz_sg(k)+
(dt/(Jz))*((wx_sg(k)*wy_sg(k))*(Jx-Jy)+NT);

        % Body Angles
        sig_apriori(1,k)=phi_sg(k)+dt*(wx_sg(k)*cos(psi_sg(k))-
wy_sg(k)*sin(psi_sg(k)));

sig_apriori(2,k)=theta_sg(k)+dt*(wx_sg(k)*sin(psi_sg(k))/cos(phi_s
g(k))+wy_sg(k)*cos(psi_sg(k))/cos(phi_sg(k))+w);

sig_apriori(3,k)=psi_sg(k)+dt*(tan(phi_sg(k))*(wx_sg(k)*sin(psi_sg
(k))+wy_sg(k)*cos(psi_sg(k)))+wz_sg(k));

        sig_apriori(1,k) = wrapToPi(sig_apriori(1,k));
        sig_apriori(3,k) = wrapToPi(sig_apriori(3,k));


        % Magnetometer Bias
        sigma = 1e-4;
        sig_apriori(7,k) = bcx_sg(k); %+ sigma*randn(1,1);
        sig_apriori(8,k) = bcy_sg(k); %+ sigma*randn(1,1);
        sig_apriori(9,k) = bcz_sg(k); %+ sigma*randn(1,1);

        % Gyro Bias
        sig_apriori(10,k) = bgx_sg(k);
        sig_apriori(11,k) = bgy_sg(k);
        sig_apriori(12,k) = bgz_sg(k);
    end

    %forecast state
    x_apriori = 0;
    x_apriori = Wm(1)*sig_apriori(:,1);
    for k = 2:size(sig_apriori,2)
        x_apriori = x_apriori+Wm(k)*sig_apriori(:,k);
    end

    %-------------------------------------------------------------
    % Covariance prior estimation
```

```
    P_apriori(:,:) =
zeros(size(sig_apriori,1),size(sig_apriori,1));
    P_apriori(:,:) = Wc(1)*(sig_apriori(:,1) -
x_apriori)*(sig_apriori(:,1) - x_apriori)';
    for k = 2:size(sig_apriori,2)
        P_apriori(:,:) = P_apriori(:,:) + Wc(k)*((sig_apriori(:,k)
- x_apriori)*(sig_apriori(:,k) - x_apriori)');
    end

    %-------------------------------------------------------------
    % Measurement sigma point
    meas_sig_points = 0;
    for k = 1:size(sig_apriori,2)
  A3 = [ cos(sig_apriori(3,k))*cos(sig_apriori(2,k)) +
sin(sig_apriori(1,k))*sin(sig_apriori(3,k))*sin(sig_apriori(2,k)),
cos(sig_apriori(1,k))*sin(sig_apriori(3,k)),
cos(sig_apriori(2,k))*sin(sig_apriori(1,k))*sin(sig_apriori(3,k))
- cos(sig_apriori(3,k))*sin(sig_apriori(2,k))

cos(sig_apriori(3,k))*sin(sig_apriori(1,k))*sin(sig_apriori(2,k))
- cos(sig_apriori(2,k))*sin(sig_apriori(3,k)),
cos(sig_apriori(1,k))*cos(sig_apriori(3,k)),
sin(sig_apriori(3,k))*sin(sig_apriori(2,k)) +
cos(sig_apriori(3,k))*cos(sig_apriori(2,k))*sin(sig_apriori(1,k))
           cos(sig_apriori(1,k))*sin(sig_apriori(2,k)),-
sin(sig_apriori(1,k)),cos(sig_apriori(1,k))*cos(sig_apriori(2,k))]
;

        % Angle and Rate Measurements
        meas_sig_points(1:3,k) = sig_apriori(1:3,k);
   meas_sig_points(4:6,k) =
sig_apriori(4:6,k)+sig_apriori(10:12,k) - x_aposteriori(10:12,i-
1);
    meas_sig_points(7:9,k) = A3*mag_orbit(:,i) +sig_apriori(7:9,k)
- x_aposteriori(7:9,i-1);

    end

    %-------------------------------------------------------------
    % Measurement sigma points estimation
    z_sig = 0;
    z_sig = Wm(1)*meas_sig_points(:,1);
    for k = 2:size(sig_apriori,2)
        z_sig = z_sig + Wm(k)*meas_sig_points(:,k);
    end

    %-------------------------------------------------------------
    % Variance and Covariance calculation
    % Measurement variance

    Pzz = 0;
    Pzz =  Wc(1)*((meas_sig_points(:,1) -
z_sig)*(meas_sig_points(:,1) - z_sig)');
    for k = 2:size(sig_apriori,2)
        Pzz = Pzz + Wc(k)*((meas_sig_points(:,k) -
z_sig)*(meas_sig_points(:,k) - z_sig)');
    end
    Pz = Pzz + R(:,:,i);

    % State-Measurement covariance
```

```
    Pxz = zeros(size(sig_apriori,1),size(z_sig,1),1);
    Pxz =  Wc(1)*((sig_apriori(:,1) -
x_apriori)*(meas_sig_points(:,1) - z_sig)');
    for k = 2:size(sig_apriori,2)
        Pxz = Pxz + Wc(k)*((sig_apriori(:,k) -
x_apriori)*(meas_sig_points(:,k) - z_sig)');
    end

    %-------------------------------------------------------------
    % Kalman Gain
    Kg(:,:,i) = Pxz/Pz;

    inno = [z(1:3,i) - z_sig(1:3);
            z(4:6,i) - z_sig(4:6) ;
            z(7:9,i) - z_sig(7:9) ];

    ind(İ) = inno' * 1/pz * inno

    if ind(i) > 23.03
        Ck = Pzz;
        Ck_hat = inno*inno';
        Sk(:,i) = max(1,diag(Ck_hat./diag(Ck)));

        for b = 1:6
            for j = 1:9
                Kg(:,:,i) = Kg(:,:,i)/Sk(j,i);
            end
        end
    end

    %-------------------------------------------------------------
    % Update Apriori state
    x_aposteriori(:,i) = x_apriori + Kg(:,:,i)*inno;

    %-------------------------------------------------------------
    % Update Apriori covariance

    P_aposteriori(:,:,i) = P_apriori(:,:) -
Kg(:,:,i)*Pz*Kg(:,:,i)';
```

CURRICULUM VITAE

Name Surname **:** Mehmet Asım Gökçay

Place and Date of Birth **:** Bakırköy/İstanbul 10.04.1995

E-Mail **:** gokcaym@itu.edu.tr

EDUCATION **:**

- **B.Sc.** **:** 2017, Istanbul Technical University, Faculty of Aeronautics and Astronautics, Astronatuical Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2017-cont : Turkish Aerospace

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- Haciyev, C. and Gokcay, M.A. (2019). Nanosatellite Attitude Estimation via Weighted Least Square Method and Error Analysis. *2019 9th International Conference on Recent Advances in Space Technologies (RAST)*, Istanbul, Turkey, June 01-04.