Comparison of Extended Kalman Filter and Unscented Kalman Filter for Flight Path Reconstruction in System Identification

“Vergleich des Extended Kalman Filters und des Unscented Kalman Filters bei der Flugbahnrekonstruktion im Zuge der Systemidentifikation”

Masterarbeit

Author: Mohit Mehndiratta
Matriculation Number: 03645448
Supervisor: Christoph Göttlicher

October 2014
Statutory Declaration

I, Mohit Mehndiratta, declare on oath towards the Institute of Flight System Dynamics of Technische Universität München, that I have prepared the present Master thesis independently and with the aid of nothing but the resources listed in the bibliography. This thesis has neither as-is nor similarly been submitted to any other university.

Garching 31 October 2014

Mohit Mehndiratta
Abstract

The Extended Kalman Filter (EKF) has become a widely used nonlinear state estimator for Flight Path Reconstruction purposes. But owing to its inherent flaw of linearizing the nonlinear system around recent state estimates, numerous new techniques have been investigated and amongst them is a type of Sigma-Point Kalman Filter also known as the Unscented Kalman Filter (UKF) and is based on the propagation of the carefully created sigma points through the nonlinear system rather than its linearization.

This thesis executes a performance evaluation of the commonly used Extended Kalman Filter with a comparatively recent Unscented Kalman Filter, regarding the Flight Path Reconstruction procedure of an aircraft model through its simulated data.

The detailed descriptions of Extended Kalman Filter and Unscented Kalman Filter are presented in the second chapter along with their derivations. Firstly, the standard Kalman Filter for linear systems is discussed, followed by its further extension to the nonlinear systems in the form of Extended Kalman Filter. Thereafter, the Unscented Transformations are illustrated and their higher precision in mean and covariance estimation over the linearization approach is verified. Additionally, the Extended and Unscented versions of the RTS smoother are derived in anticipation of further improvement of the estimates.

The mentioned techniques are then applied to an example tracking problem of a Re-Entry vehicle and a superior performance of Unscented Kalman Filter over the Extended Kalman Filter was witnessed in state estimation of the concerned dynamic system.

Finally, the two filters were implemented for the Flight Path Reconstruction of a simulation model, using data obtained from the ‘X-Plane’ and the attained results were used to carry out a performance comparison between the two filters, based on various considered grounds. The shortcomings of each filter, as evident from the results, were outlined and discussed in detail. Moreover, for the given dynamic model, the Unscented Kalman Filter achieves better estimates only in case of degraded initial conditions whereas the Extended Kalman Filter delivers enhanced results for all the other factors.
# Table of Contents

List of Figures .................................................................................................................. viii
List of Tables ..................................................................................................................... x
Table of Acronyms .............................................................................................................. x
Table of Symbols ............................................................................................................... xi
1 Introduction .................................................................................................................... 1
2 Theory ............................................................................................................................. 3
   2.1 Flight Path Reconstruction (FPR) ........................................................................... 3
   2.2 The Kalman Filter ..................................................................................................... 5
   2.3 The Extended Kalman Filter ................................................................................... 6
      2.3.1 Derivation of Equations .................................................................................. 7
      2.3.2 Extended Rauch-Tung-Striebel Smoother ....................................................... 14
      2.3.3 Problems with EKF ....................................................................................... 16
   2.4 Unscented Transformation ....................................................................................... 17
      2.4.1 Unscented Kalman Filter ............................................................................... 28
      2.4.2 Unscented Rauch, Tung and Striebel Smoother ............................................. 33
   2.5 The Model ................................................................................................................ 41
      2.5.1 Dynamics of the System ................................................................................. 41
   2.6 Re-entry Vehicle Example ...................................................................................... 46
      2.6.1 Problem description ....................................................................................... 46
      2.6.2 Results ............................................................................................................. 46
3 Results ............................................................................................................................ 50
   3.1 Computational Time ............................................................................................... 58
   3.2 Poor Initial Guess ................................................................................................... 59
   3.3 Un-tuned Noise Covariance Matrices .................................................................... 64
   3.4 Lower Measuring Rates ....................................................................................... 70
   3.5 Lower Sampling Rates ........................................................................................... 71
4 Documentation ............................................................................................................... 77
   4.1 Detailed Class Description ..................................................................................... 78
      4.1.1 The FPR class .................................................................................................. 78
      4.1.2 The PROBLEM class .................................................................................... 80
      4.1.3 The UKF class ................................................................................................ 81
      4.1.4 The EKF class ................................................................................................ 82
   4.2 Other Auxiliary functions ....................................................................................... 83
      4.2.1 Function UKF_FUN_AUG_UNSTORED ....................................................... 83
## Contents and Lists

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.2 UKF_SMOOTH</td>
<td>84</td>
</tr>
<tr>
<td>4.2.3 Function EKF_FUN</td>
<td>84</td>
</tr>
<tr>
<td>4.2.4 EKF_SMOOTH</td>
<td>84</td>
</tr>
<tr>
<td>4.2.5 Function NAME_CHECK</td>
<td>84</td>
</tr>
<tr>
<td>4.3 Re-entry Vehicle Example</td>
<td>86</td>
</tr>
<tr>
<td>4.3.1 Implementation</td>
<td>86</td>
</tr>
<tr>
<td>4.3.2 Set up for application</td>
<td>86</td>
</tr>
<tr>
<td>Observation</td>
<td>87</td>
</tr>
<tr>
<td>5 Conclusions and perspective</td>
<td>88</td>
</tr>
<tr>
<td>6 References</td>
<td>I</td>
</tr>
<tr>
<td>A. Appendix A – Fundamental Derivations</td>
<td>II</td>
</tr>
<tr>
<td>A.1 Relation between Continuous and Discrete Time Linear Dynamic Systems</td>
<td>II</td>
</tr>
<tr>
<td>A.2 The Linear Kalman Filter</td>
<td>III</td>
</tr>
<tr>
<td>A.3 Solving the System Equations</td>
<td>VI</td>
</tr>
<tr>
<td>A.4 Probability density function (pdf) of a random variable (RV)</td>
<td>VII</td>
</tr>
<tr>
<td>A.5 Equivalence of the Kalman Gain Matrix Expression in EKF and UKF</td>
<td>VIII</td>
</tr>
<tr>
<td>B. Appendix B – Additional Figures</td>
<td>X</td>
</tr>
<tr>
<td>B.1 Smoother Results</td>
<td>X</td>
</tr>
<tr>
<td>B.2 Poor Initial Guess</td>
<td>XVI</td>
</tr>
<tr>
<td>B.3 Results For Variation in R</td>
<td>XX</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1-1: TSM representation ........................................................................................................ 1
Figure 2-1: Block diagram of UT taken from (Hayking, 2001) ..................................................... 20
Figure 2-2: Estimated States ........................................................................................................... 47
Figure 2-3: Reconstructed Output .................................................................................................. 47
Figure 2-4: Absolute estimation error ............................................................................................. 47
Figure 2-5: Average estimation error over 50 Monte Carlo Simulation .......................................... 48
Figure 2-6: Average Estimation error for various time steps ......................................................... 49
Figure 2-7: Average mean-square error over various linearization times ...................................... 49
Figure 3-1: Inputs – Linear accelerations ....................................................................................... 51
Figure 3-2: Inputs – Rotational rates ............................................................................................... 51
Figure 3-3: Inputs – Rotational accelerations .................................................................................. 52
Figure 3-4: Estimated States – Position ........................................................................................ 52
Figure 3-5: Absolute Estimation Error - Position .......................................................................... 53
Figure 3-6: Estimated States – Kinematic velocities ..................................................................... 53
Figure 3-7: Absolute Estimation Error – Kinematic velocities ...................................................... 54
Figure 3-8: Estimated States – Attitude angles (Euler angles) ...................................................... 54
Figure 3-9: Absolute Estimation Error – Attitude angles .............................................................. 55
Figure 3-10: Estimated States – Wind velocities ............................................................................ 55
Figure 3-11: Absolute Estimation Error – Wind velocities ............................................................ 56
Figure 3-12: Reconstructed Outputs – Position ............................................................................ 56
Figure 3-13: Reconstructed Outputs – Kinematic velocities ........................................................ 57
Figure 3-14: Reconstructed Outputs – Air data sensor quantities ................................................ 57
Figure 3-15: States – Position with varying Initial conditions ....................................................... 60
Figure 3-16: States – Kinematic Velocities with varying Initial conditions ................................... 60
Figure 3-17: States – Attitude angles with varying Initial conditions ............................................ 61
Figure 3-18: States – Wind Velocities with varying Initial conditions .......................................... 61
Figure 3-19: Outputs – Position with varying Initial conditions .................................................. 62
Figure 3-20: Outputs – Kinematic Velocities with varying Initial conditions ............................. 62
Figure 3-21: Outputs – Air Data quantities with varying Initial conditions ................................. 63
Figure 3-22: Average MSE for percentage variation in Initial conditions ..................................... 63
Figure 3-23: States – Position with varying $Q$ .............................................................................. 65
Figure 3-24: States – Kinematic Velocities with varying $Q$ .......................................................... 65
Figure 3-25: States – Attitude angles with varying $Q$ ................................................................. 66
Figure 3-26: States – Wind Velocities with varying $Q$ .................................................................. 66
List of Tables

Table 2-1: Computational time [sec] .................................................................48
Table 2-2: Mean-square estimation error ..........................................................48
Table 3-1: Computational time [sec] ..................................................................58
Table 3-2: Average mean-square estimation error .............................................59
Table 4-1: Properties of class FPR with protected access .................................78
Table 4-2: Properties of class FPR with public access .......................................78
Table 4-3: Methods of class FPR ......................................................................79
Table 4-4: Properties of class PROBLEM with protected access .................80
Table 4-5: Methods of class PROBLEM ............................................................80
Table 4-6: Properties of class UKF with protected access ...............................81
Table 4-7: Methods of class UKF ....................................................................82
Table 4-8: Properties of class EKF with protected access ...............................82
Table 4-9: Methods of class EKF ....................................................................83
Table 4-10: Function description of UKF_FUN_AUG_UNSTORED ...............83
Table 4-11: Function description of UKF_SMOOTH ........................................84
Table 4-12: Function description of EKF_FUN ................................................84
Table 4-13: Function description of EKF_SMOOTH .........................................84
Table 4-14: Function description of NAME_CHECK ........................................85
Table 4-15: Data provided ..............................................................................86
Table 4-16: Inputs by user ..............................................................................87

Table of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSPIF</td>
<td>Backward Sigma-Point Information Filter</td>
</tr>
<tr>
<td>EKF</td>
<td>Extended Kalman Filter</td>
</tr>
</tbody>
</table>
ERTS | Extended Rauch-Tung-Striebel Smoother
---|---
FER | Filter Error Method
FPR | Flight Path Reconstruction
GRV | Gaussian Random Variable
IMU | Inertial Measurement Unit
INS | Inertial Navigation System
KF | Kalman Filter
LKF | Linearised Kalman Filter
MSE | Mean-square estimation error
NB | Nose Boom
OEM | Output Error Method
R-K | Runge-Kutta
RV | Random variables
TSM | Two-step method
UKF | Unscented Kalman Filter
URTS | Unscented Rauch-Tung-Striebel Smoother
UT | Unscented Transform

Table of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a^\text{INS}K) (^{\text{I}})</td>
<td>(m/s^2)</td>
<td>Kinematic acceleration vector given at INS location (sensor reference point) according to FSD convention</td>
</tr>
<tr>
<td>(A_k)</td>
<td>varying</td>
<td>Linearized system matrix</td>
</tr>
<tr>
<td>(B_k)</td>
<td>varying</td>
<td>Linearized input matrix</td>
</tr>
<tr>
<td>(C_k)</td>
<td>varying</td>
<td>Linearized output matrix</td>
</tr>
<tr>
<td>(D_k)</td>
<td>varying</td>
<td>Linearized feed-through matrix</td>
</tr>
<tr>
<td>(D_xf)</td>
<td>--</td>
<td>Differential operator</td>
</tr>
<tr>
<td>(\hat{e}_k)</td>
<td>varying</td>
<td>Estimation error</td>
</tr>
<tr>
<td>(E_k)</td>
<td>varying</td>
<td>Linearized output process noise matrix</td>
</tr>
<tr>
<td>(f(t, x, u, w))</td>
<td>varying</td>
<td>Dynamic system equation</td>
</tr>
<tr>
<td>(F_d, F_k)</td>
<td>varying</td>
<td>Linearized process noise distribution matrix</td>
</tr>
<tr>
<td>(g)</td>
<td>(m/s^2)</td>
<td>Gravitational acceleration</td>
</tr>
<tr>
<td>(G_d, G_k)</td>
<td>varying</td>
<td>Linearized measurement noise distribution matrix</td>
</tr>
<tr>
<td>(h(t, x, u, v))</td>
<td>varying</td>
<td>Dynamic output equation</td>
</tr>
<tr>
<td>((h^G)_{\text{WGS84}})</td>
<td>(m)</td>
<td>Altitude at center of gravity</td>
</tr>
<tr>
<td>(I)</td>
<td>varying</td>
<td>Identity matrix</td>
</tr>
<tr>
<td>(I_k)</td>
<td>varying</td>
<td>Information matrix</td>
</tr>
</tbody>
</table>
### Contents and Lists

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_k$</td>
<td>varying Kalman filter gain matrix</td>
</tr>
<tr>
<td>$M_k$</td>
<td>varying Projected gain matrix for RTS</td>
</tr>
<tr>
<td>$P_{x}$, $P_{k}$</td>
<td>varying State covariance matrix</td>
</tr>
<tr>
<td>$P_{xy}$</td>
<td>varying Cross covariance matrix</td>
</tr>
<tr>
<td>$P_{yy}$</td>
<td>varying Output covariance matrix</td>
</tr>
<tr>
<td>$Q_k$</td>
<td>varying Process noise covariance matrix</td>
</tr>
<tr>
<td>$R_k$</td>
<td>varying Measurement noise covariance matrix</td>
</tr>
<tr>
<td>$u_k$</td>
<td>varying Input vector</td>
</tr>
<tr>
<td>$w(t), w_k$</td>
<td>varying Process noise vector</td>
</tr>
<tr>
<td>$W^{(m)}$</td>
<td>$-$ Weight of sigma point for computing mean</td>
</tr>
<tr>
<td>$W^{(c)}$</td>
<td>$-$ Weight of sigma point for computing covariance matrix</td>
</tr>
<tr>
<td>$\hat{x}_k$</td>
<td>varying Estimated state vector</td>
</tr>
<tr>
<td>$\hat{y}_k$</td>
<td>varying Estimated output vector</td>
</tr>
<tr>
<td>$z_k$</td>
<td>varying Measurements at time $k$</td>
</tr>
</tbody>
</table>

### Greek Letters

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>$rad$</td>
<td>Angle of attack</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$rad$</td>
<td>Angle of side slip</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>$-$</td>
<td>Composite scaling parameter for sigma points</td>
</tr>
<tr>
<td>$(\lambda^G)_{\text{WGS84}}$</td>
<td>$deg$</td>
<td>Longitudinal position coordinate of the center of gravity</td>
</tr>
<tr>
<td>$(\Phi^G)_{\text{WGS84}}$</td>
<td>$deg$</td>
<td>Latitudinal position coordinate of the center of gravity</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>$-$</td>
<td>Secondary scaling parameter for sigma points</td>
</tr>
<tr>
<td>$\nu(t), \nu_k$</td>
<td>varying</td>
<td>Measurement noise vector</td>
</tr>
<tr>
<td>$\rho$</td>
<td>$kg/m^3$</td>
<td>Air density</td>
</tr>
<tr>
<td>$\Phi_k$</td>
<td>varying</td>
<td>State transition matrix</td>
</tr>
<tr>
<td>$\Psi_k$</td>
<td>varying</td>
<td>Discrete input matrix</td>
</tr>
<tr>
<td>$\Psi_{\text{euler}}$</td>
<td>$rad$</td>
<td>Euler attitude angles vector</td>
</tr>
<tr>
<td>$(\omega^K_{r})$</td>
<td>$rad/s$</td>
<td>Kinematic rotational rates of the aircraft, based on FSD conventions</td>
</tr>
<tr>
<td>$\chi_i$</td>
<td>varying</td>
<td>Sigma point vector</td>
</tr>
<tr>
<td>$\chi^{a}_{k-1}$</td>
<td>varying</td>
<td>Augmented form of sigma point vector</td>
</tr>
<tr>
<td>$\chi^{w}_{k-1}$</td>
<td>varying</td>
<td>Sigma point vector of the process noise</td>
</tr>
<tr>
<td>$\chi^{x}_{k-1}$</td>
<td>varying</td>
<td>Sigma point vector of the states</td>
</tr>
<tr>
<td>$\chi^{y}_{k-1}$</td>
<td>varying</td>
<td>Sigma point vector of the measurement noise</td>
</tr>
<tr>
<td>$Y_i$</td>
<td>varying</td>
<td>Transformed sigma point vector</td>
</tr>
</tbody>
</table>

### Indices

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>Variable related to linear acceleration</td>
</tr>
<tr>
<td>$A$</td>
<td>Aerodynamic quantities</td>
</tr>
<tr>
<td>$b$</td>
<td>Backward estimate</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>$B$</td>
<td>Body-fixed reference frame</td>
</tr>
<tr>
<td>$baro$</td>
<td>Variable related to barometric measurement</td>
</tr>
<tr>
<td>$d$</td>
<td>Discrete time variable</td>
</tr>
<tr>
<td>$euler$</td>
<td>Euler angles</td>
</tr>
<tr>
<td>$E$</td>
<td>Earth-fixed reference frame</td>
</tr>
<tr>
<td>$f$</td>
<td>Forward estimates</td>
</tr>
<tr>
<td>$G$</td>
<td>Center of gravity as a reference point</td>
</tr>
<tr>
<td>$GPS, meas$</td>
<td>GPS measurements</td>
</tr>
<tr>
<td>$I$</td>
<td>Inertial reference frame</td>
</tr>
<tr>
<td>$IAS$</td>
<td>Indicated airspeed</td>
</tr>
<tr>
<td>$INS, meas$</td>
<td>INS measurements</td>
</tr>
<tr>
<td>$k$</td>
<td>Time instant</td>
</tr>
<tr>
<td>$K$</td>
<td>Kinematic quantities</td>
</tr>
<tr>
<td>$O$</td>
<td>North-East-Down reference frame</td>
</tr>
<tr>
<td>$s$</td>
<td>Smoothed estimate</td>
</tr>
<tr>
<td>$TAS$</td>
<td>True airspeed</td>
</tr>
<tr>
<td>$W$</td>
<td>Wind quantities</td>
</tr>
<tr>
<td>$\omega$</td>
<td>Variable related to rotational velocity</td>
</tr>
<tr>
<td>$\dot{\omega}$</td>
<td>Variable related to rotational acceleration</td>
</tr>
</tbody>
</table>
1 Introduction

The technological advancement during the last few decades in modern computing and flight test techniques has made a remarkable impact on implementation and theory of Aircraft parameter identification. The primary objectives of various flight test programs are to obtain airworthiness certification and to estimate aircraft’s performance characteristics and stability and control derivatives using linearized equations of motion. Numerous techniques for aircraft parameter identification, both in time and frequency domain, have been applied in the past and amongst them some are Maximum-Likelihood methods, Filter Error method, Two-Step method (TSM).

The Two-Step method is one of the prominent techniques that have been proved efficient for aircraft system identification purpose. According to (J.A. Mulder, 1999) in two-step method, the state-parameter identification problem comprises of a nonlinear state estimation or state reconstruction problem and a consequent parameter identification problem. In literature, the first step of TSM is often known as ‘Flight Path Reconstruction (FPR)’. This report is entirely focused on Flight path reconstruction step of the TSM approach. A diagrammatic representation of TSM approach is given below in Figure 1-1.

![Figure 1-1: TSM representation](image)

The significance of the flight path reconstruction (or the process of state estimation) is not only limited to the aerial vehicles, but also to any dynamic system, which operates with uncertainties in the process and measurement models. However, the accuracy of the state estimates depend entirely on the state estimator employed. One of the state estimator for nonlinear estimation, which had been extensively used in the past and even today, is the Extended Kalman Filter. But due to its inherent limitation that it is based on the linearization of the nonlinear system, some more new techniques have been successfully implemented and amongst them is the Sigma Point Filtering (or Unscented Kalman Filtering), which is based on the propagation of the sigma points rather than linearizing the system. Hence the motivation of this report is to compare the performance of the frequently used Extended Kalman Filter with the performance of a comparatively newer approach of the Unscented Kalman Filtering, over various aspects in flight path reconstruction of a nonlinear system.

Due to the lack of the real flight data, the test flight data was generated using a simulation software ‘X-Plane’ for the aircraft model of ‘B-777’. The main advantage was the availability of the true states and outputs for an efficient later comparison. All the necessary maneuvers in both the planes (longitudinal and lateral-directional) were performed to excite the aircraft fundamental modes. The purpose was to observe whether the states are correctly estimated, along with the proper tracking of the scattered measurements by the reconstructed outputs.
For the mentioned design conditions (using all the optimum parameters), a good fit was observed between the estimated states and the true states by both the filters (EKF and UKF). But for the various considered aspects, both of them show distinct performances, which is discussed in the corresponding sections.

A thorough discussion about each topic was focused. The derivation of all the necessary equations is provided in the respective sections. For further enhancement of the fundamental in few topics, some basics are reviewed in the appendix. The main aim was to be as explicit as possible, so that any engineering background personnel should be able to have an insight into the topic without looking up into any other information source.

The work is broadly distributed among five chapters, starting with a short description about the topic in chapter 1 (Introduction). Chapter 2 (Theory) forms the main theoretical part of this report, explaining all the theoretical concepts in an exhaustive manner. First, it explores the Flight Path Reconstruction step of the Two-Step method (section 2.1). Then the Linear Kalman Filter (section 2.2) is discussed without derivation, but a brief algorithm is presented. Afterwards, the Extended Kalman Filter (section 2.3) is explained and all the necessary equations are given; accompanied by the discussion of the Extended RTS smoother, along with the derivation of its equations. Later, the Unscented Transforms (section 2.4) are enlightened and its application to an extension, as the Unscented Kalman Filter is shown in section 2.4.1; followed by the derivation for the unscented version of the RTS smoother. At the end of this chapter, a brief description about the considered model is presented (section 0).

All the obtained results are explained in chapter 3 (Results). Different observed behavior of both the filters, for all the considered aspects are mentioned, along with their reasoning, in various sections of this chapter.

The software description is done in chapter 4 (Documentation). The object oriented procedure used in programming is thoroughly explained in section 4.1 along with the description of the various created classes. Some other auxiliary functions are mentioned in section 4.2 with their working and purpose of creation. Later in this chapter, one example problem is treated (section 0) and the results are discussed along with the obtained tables and figures.

Finally, the drawn conclusions are provided in chapter 5 (Conclusions and perspective).

Further material containing some fundamental derivations are given in appendix A together with some additional plots, which are presented in appendix B.
2 Theory

2.1 Flight Path Reconstruction (FPR)

During a flight test, quantities like ‘rate of climb’ can be measured directly (also, quantities like ‘natural frequencies’ and ‘damping ratios’ of the Eigen modes can be obtained by a later analysis of the measurements) while some are rather difficult to measure (for example ‘angle of attack’) and may lead to problems if they are tried to measure directly with the available instruments like with a pitot-tube and gyros. This forms the basis for flights path reconstruction. In simple terms, it is a technique of accurately determining or reconstructing the time history of the aircraft’s position, attitude, velocities and rotational rates (or states to be precise) using the measurements made during a flight and also assuring consistency between the aircraft kinematic equations and the flight data. As mentioned in (J.A. Mulder, 1999), “The term estimation is based on the measurements up to and including the present time whereas, for reconstruction all measurements of the complete flight test are available and used to calculate the state vector”. From this, it is understood that the reconstruction can only be used for an off-line data analysis and furthermore, as the reconstruction process is based on past as well as future measurements, therefore it is often more accurate than the estimation process.

The flight path reconstruction utilizes the redundancy present in the available measurements from inertial and air data sensors as the dynamic flight data if often susceptible to bias and scale factor errors along with the presence of process and measurement noise; thus FPR gives a best estimate for the states along with an estimate of scale factors and bias errors. In a nutshell, FPR involves in the reconstruction of states and sensors errors (if the state vector is augmented with additional states) using the kinematic equations that relate, position, velocity and acceleration including scale and bias errors, as unknown parameter.

Following the FPR, which forms the first step of TSM, the reconstructed states are used for aircraft dynamic modeling and subsequently, several performance, stability and control characteristics can then be obtained directly from the aerodynamic model. This is the second step of the TSM approach and one of the renowned methods used is the ‘Output Error method (OEM)’.

The output error method is based on the maximum likelihood principles in which the parameters are iteratively adjusted to minimize the error between the measured variables (system’s output) and estimated response (model’s prediction). The model which is considered for OEM is the following continuous-discrete time dynamic system (it is a form of hybrid system which includes a continuous system with discrete measurements).

\[
\dot{x}(t) = f(t, x(t), u(t)); \quad x(t = 0) = x_0 \\
y(t) = h(t, x(t), u(t)) \\
z_k = y_k + G_d v_k; \quad z_k = z(t_k); \quad t_k = t_0 + k\Delta t \\
x \in \mathbb{R}^l; \quad y, z_k \in \mathbb{R}^m; \quad u \in \mathbb{R}^n; \quad v_k \in \mathbb{R}^{n_v}; \quad G_d \in \mathbb{R}^{m \times n_v}
\]
2 Theory

Where,

\[ x \] is the state vector
\[ u \] is the input vector
\[ y \] is the output vector
\[ z_k \] is the discrete measurement vector

Also, \( f \) and \( h \) represent the nonlinear system and output functions respectively.

The above mentioned system in deterministic, as the only source of noise present is the discrete measurement noise \( \nu_k \), which is assumed to be an additive Gaussian white noise with \( E[\nu(t)] = 0 \). The noise in measurements represent a certain degree of uncertainty in their knowledge. Thus, the time history of the states can be obtained by directly integrating the system equation using the given initial condition \( x_0 \) and corresponding inputs \( u(t) \). Further description of this method is beyond the scope of this report and one can refer (Jategaonkar, 2006) for detailed implementation.

The maximum likelihood method (like OEM) as described above, are efficient in evaluating the states of a system; given that the system is deterministic i.e., besides the measurement noise, there should not be any other uncertainties entering the dynamic system equations. But for stochastic systems, these methods often yield poor results on account of the presence of an additional noise source, that is known as the ‘process noise’, which signifies the presence of an uncertainty in the system model and can be seen as ‘Turbulence’ for example. The stochastic system considered, is of the form

\[
\dot{x}(t) = f(t, x(t), u(t)) + Fw(t); \quad x(t = 0) = x_0
\]

\[
y(t) = h(t, x(t), u(t))
\]

\[
z_k = y_k + G_d v_k
\]

\[ x \in \mathbb{R}^L; \quad y, z_k \in \mathbb{R}^n; \quad u \in \mathbb{R}^n; \quad w \in \mathbb{R}^{nw}; \quad v_k \in \mathbb{R}^{nv}; \quad F \in \mathbb{R}^{L \times nw}; \quad G_d \in \mathbb{R}^{m \times nv}\]

Where, \( w(t) \) and \( v(t) \) are the assumed, additive Gaussian process noise and additive Gaussian measurement noise respectively. Also, \( E[w(t)] = 0 \) and \( E[v(t)] = 0 \).

The problem in estimating the states of a stochastic system is that the system equation cannot be directly integrated over time even with known initial condition \( x_0 \) and inputs. This is due to the presence of a random variable (process noise) in the system’s dynamic equations and thus leads to the requirement of statistical methods for its state estimation. Most of the statistical state estimators (often called ‘filters’) are based on the ‘Bayesian formulation’ that utilizes ‘probability density functions (pdf)’ in describing the states and measured variables. The goal of these state estimators is to perform the ‘optimal filtering’, which refers to a methodology that estimates the states of a time varying system using the available noisy measurements. One of the widely used state estimator based on this approach is the ‘Kalman Filter’.

There is another technique in which the flight path reconstruction becomes a joint state-parameter estimation problem and is often known as ‘Filter Error method (FEM)’ in the literature. This approach estimates the states of a system along with the estimation of the
unknown system parameters, which is usually done in the second step of the previously mentioned two-step method. On account of high algorithmic complexities associated with this method, like divergence issues, it has not been used extensively for aircraft parameter identification as for some problems, superior results were obtained by a simple preprocessing of the data, followed by the parameter estimation using the output error method, without facing much algorithmic difficulties as seen in case of the filter error method.

2.2 The Kalman Filter

The aim of Kalman filtering is to have an unbiased, minimum variance state estimator. It is a two-step procedure which has a ‘prediction step’ and a ‘correction update’. The purpose of prediction step is to estimate the states, based on the knowledge about the system. As the process noise is not measurable, hence only the deterministic part of the states are propagated in the prediction step and later these erroneous predictions are corrected at regular intervals during the correction update with the measured data. Considering a Linear time-invariant, continuous time dynamic system

$$\dot{x}(t) = Ax(t) + Bu(t) + Fw(t)$$
$$z(t) = Cx(t) + Gv(t)$$

(2-3)

The discrete time equivalent of the above equation is (see section 0 for detailed derivation)

$$x_{k+1} = \Phi(\Delta t)x_k + \Psi(\Delta t) \cdot Bu_k + \Psi(\Delta t) \cdot F_d w_k$$
$$z_k = Cx_k + \frac{1}{\sqrt{\Delta t}} G v_k = Cx_k + G_d v_k$$

(2-4)

Where,

$$\Phi_k = e^{A_k \Delta t} = \sum_{i=0}^{\infty} \frac{(A_k \Delta t)^i}{i!}$$
(2-5)

$$\Psi_k = \int_0^{\Delta t} e^{A_k \tau} d\tau = \sum_{i=1}^{\infty} \frac{A_k^{i-1} \Delta t^i}{i!}$$
(2-6)

The complete derivation of the standard Kalman filter for a linear system is omitted here, but because of its significance, it is provided in appendix A.2. Nevertheless, a brief algorithm of the Kalman filter for linear discrete system is mentioned below. Here, in representation of state vector $\hat{x}_k$, the superscript ‘ $-\cdot$ ’ means that the estimation is an ‘uncorrected prediction’ or a ‘priori estimate’ whereas the subscript represents the time instant, $k$. On the contrary, superscript ‘ $+\cdot$ ’ would mean that estimation is a ‘corrected’ or ‘a posteriori estimate’ with corresponding measurement update. This nomenclature is followed throughout the report.
1. The \( L \) dimensional discrete time dynamic system (\( \Delta t \) is removed for readability)

\[
\begin{align*}
    x_{k+1} &= \Phi x_k + \Psi \cdot Bu_k + \Psi \cdot F_d w_k \\
    z_k &= Cx_k + G_d v_k \\
    w_k &\sim (0, Q_k) \\
    v_k &\sim (0, R_k)
\end{align*}
\]

2. Initialization

\[
\begin{align*}
    \hat{x}_0^+ &= E(x_0) \\
    P_0^+ &= E [(x_0 - \hat{x}_0^+) (x_0 - \hat{x}_0^+)^T]
\end{align*}
\]

3. Prediction

From time \( k = 1, ..., N-1 \)

\[
\begin{align*}
    \hat{x}_{k+1}^- &= \Phi \hat{x}_k^+ + \Psi \cdot Bu_k \\
    P_{k+1}^- &= \Phi P_k^+ \Phi^T + \Psi F_d Q_k F_d^T \Psi^T
\end{align*}
\]

4. Correction update

\[
\begin{align*}
    K_{k+1} &= P_{k+1}^- C \left( CP_{k+1}^- C^T + G_d R_k G_d^T \right)^{-1} \\
    \hat{x}_{k+1}^+ &= \hat{x}_{k+1}^- + K_{k+1} [z_k - C \hat{x}_{k+1}^-] \\
    P_{k+1}^+ &= (I - K_{k+1} C) P_{k+1}^-
\end{align*}
\]

It may be worth noting that the Kalman filter behaves as an optimal state estimator given that the state transition model and measurements are both linear and the present noise is Gaussian.

As known from experience, perfectly linear systems seldom exist; that is, in reality almost all the systems are nonlinear. And the problem lies in the fact that still the optimum solution only exist for some particular nonlinear systems and there is no general solution available (as of yet) like in existence for the linear estimation. However, many suboptimal solutions had been found for nonlinear systems that lead to descent results and some of them are Linearized Kalman Filter (LKF), Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF). Here, only Extended Kalman Filter and Unscented Kalman Filter are discussed in detail.

2.3 The Extended Kalman Filter

In nonlinear optimization, the nonlinear system is linearized about the nominal states, controls, outputs and noise values. For instance, if the system equations represent the dynamics of a missile then the nominal control would be those control surface deflections, given to achieve the states (position, velocity and attitude) and output, required to be on the planned mission trajectory to hit the target. In Linearized Kalman filter, deviation of true states from the nominal states is estimated instead of estimating the true states, which is the basis for Extended Kalman filter. Another implementation difference of Linearized Kalman filter from Extended Kalman filter is that, for EKF the nonlinear system is linearized about the most recent prediction of the Kalman filter instead of the nominal states and controls because in reality, the latter are very difficult to measure.
The fundamental goal behind EKF is to use the whole nonlinear dynamic system in state $\dot{x}_k$ propagation, whereas the state error covariance matrix $P_k$ and measurement updates are propagated in the form of Linearized system matrices. Firstly, the equations for the system with feed-through are derived, which is followed by an algorithmic representation of equations for the standard EKF formulation.

### 2.3.1 Derivation of Equations

The EKF algorithm presented is based on the $L$ dimensional continuous-discrete time dynamic system with Feed-through and the system equations are of the form

$$
\dot{x} = f(x(t), u(t), w(t)) \\
z = h(x(t), u(t), w(t), v(t))
$$

Also $w(t)$ and $v(t)$ are the assumed additive white Gaussian process and measurement noise, which are defined as

$$
w(t) \sim (0, Q(t)) \\
v(t) \sim (0, R(t))
$$

#### Linearization of dynamic equations

$A_k, B_k$ and $F_k$ are linearized system matrix, input matrix and process noise distribution matrix respectively, which represents the linearization of system dynamics about current state estimate $\dot{x}_k^k$.

$$
A_k = \left. \frac{\partial f(x(t), u(t), w(t))}{\partial x} \right|_{x = \dot{x}_k^k, u = u_k} \\
B_k = \left. \frac{\partial f(x(t), u(t), w(t))}{\partial u} \right|_{x = \dot{x}_k^k, u = u_k} \\
F_k = \left. \frac{\partial f(x(t), u(t), w(t))}{\partial w} \right|_{x = \dot{x}_k^k, u = u_k}
$$

The inputs (at discrete sampling instant $k$ with measured and stochastic part) are taken to be of the form

$$
u_k = \tilde{u}_k + u_{\text{meas}}
$$

With $E[\tilde{u}_k] = 0$ and $E[\tilde{u}_k \tilde{u}_k^T] = Q_{\text{meas}}$

The stochastic part of the input $\tilde{u}_k$ and the process noise $w_{\text{proc}}$ are combined to give total process noise as,

$$
w = \begin{bmatrix} \tilde{u} \\ w_{\text{proc}} \end{bmatrix}
$$

Therefore, equation (2-16) can be simplified as,
\[ F_k = \begin{bmatrix} \frac{\partial f(x(t), u(t), w(t))}{\partial \tilde{u}} \\ \frac{\partial f(x(t), u(t), w(t))}{\partial \tilde{u}} \end{bmatrix} \bigg|_{x=\tilde{x}_k^+, \ u=u_k} \begin{bmatrix} \frac{\partial f(x(t), u(t), w(t))}{\partial w_{proc}} \\ \frac{\partial f(x(t), u(t), w(t))}{\partial w_{proc}} \end{bmatrix} \bigg|_{x=\tilde{x}_k^+, \ u=u_k} \]  

Equation (2-19)

Also, for any arbitrary vector function \( g \),

\[ \frac{\partial g}{\partial \tilde{u}} = \frac{\partial g}{\partial u} \frac{\partial u}{\partial \tilde{u}} = \frac{\partial g}{\partial u} \left( \frac{\partial u_{meas}}{\partial \tilde{u}} + \frac{\partial \tilde{u}}{\partial \tilde{u}} \right) = \frac{\partial g}{\partial u} (0 + I) = \frac{\partial g}{\partial u} \]  

Equation (2-20)

Substituting this result in the previous equation

\[ F_k = [B_k \quad \bar{F}_k] \]  

Equation (2-21)

As linearization is done at \( \tilde{x}_k^+ \), the state transition matrix \( \Phi_k \) and the discrete input matrix \( \Psi_k \) are time dependent explicitly.

\[ \Phi_k = e^{A_k \Delta t} = \sum_{i=0}^{\infty} \frac{(A_k \Delta t)^i}{i!} \]  

Equation (2-22)

\[ \Psi_k = \int_0^{\Delta t} e^{A_k \tau} d\tau = \sum_{i=1}^{\infty} \frac{A_k^{-1} \Delta t^i}{i!} \]  

Equation (2-23)

On the other hand, output matrix \( C_k \), feed-through \( D_k \) and noise distribution matrices \( E_k \) and \( G_k \) are linearization of the dynamics at most current prediction \( \tilde{x}_{k+1}^- \).

\[ C_k = \frac{\partial h(x(t), u(t), w(t), v(t))}{\partial x} \bigg|_{x=\tilde{x}_{k+1}^-, \ u=u_{k+1}} \]  

Equation (2-24)

\[ D_k = \frac{\partial h(x(t), u(t), w(t), v(t))}{\partial u} \bigg|_{x=\tilde{x}_{k+1}^+, \ u=u_{k+1}} \]  

Equation (2-25)

\[ E_k = \frac{\partial h(x(t), u(t), w(t), v(t))}{\partial w} \bigg|_{x=\tilde{x}_{k+1}^+, \ u=u_{k+1}} \]  

Equation (2-26)

Using the definition of total process noise of equation (2-18) the above output process noise distribution matrix can be written as

\[ E_k = \begin{bmatrix} \frac{\partial h(x(t), u(t), w(t), v(t))}{\partial \tilde{u}} \\ \frac{\partial h(x(t), u(t), w(t), v(t))}{\partial \tilde{u}} \end{bmatrix} \bigg|_{x=\tilde{x}_{k+1}^+, \ u=u_{k+1}} \begin{bmatrix} \frac{\partial h(x(t), u(t), w(t), v(t))}{\partial w_{proc}} \\ \frac{\partial h(x(t), u(t), w(t), v(t))}{\partial w_{proc}} \end{bmatrix} \bigg|_{x=\tilde{x}_{k+1}^+, \ u=u_{k+1}} \]  

Equation (2-27)

Again using the result of equation (2-20)

\[ E_k = [D_k \quad \bar{E}_k] \]  

Equation (2-28)
Also,

$$G_k = \frac{\partial h(x(t), u(t), w(t), v(t))}{\partial v} \bigg|_{x=x_{k+1}, u=u_{k+1}}$$  \hspace{1cm} (2-29)$$

As said before for the considered dynamic system, measurement noise is assumed to be additive. Hence

$$G_k = I$$  \hspace{1cm} (2-30)$$

The only difference of the present formulation (with feed-through), from the standard EKF formulation (without the feed-through) as given in (Simon, 2006), is that for the latter case, the feed-through matrix \(D_k\) and \(E_k\) disappear (that is \(D_k = E_k = 0\)) in the above equations. For one’s consideration, a brief algorithm consisting all the necessary equations of the standard EKF formulation is presented at the end of the present section.

After using the result obtained for continuous to discrete transformation (can be seen in appendix A.1) and substituting equations (2-21) and (2-17) in that, the linearized system becomes,

$$x_{k+1} = \Phi_k x_k + \Psi_k B_k u_k + \Psi_k F_k w_k$$

$$x_{k+1} = \Phi_k x_k + \Psi_k B_k u_{meas,k} + \Psi_k (B_k \bar{u}_k + \bar{F}_k w_{proc,k})$$  \hspace{1cm} (2-31)$$

$$z_{k+1} = C_{k+1} x_{k+1} + D_{k+1} u_{meas,k+1} + E_{k+1} w_{k+1} + G_{k+1} v_{k+1}$$  \hspace{1cm} (2-32)$$

The resulting estimation errors at different times are represented as,

$$\hat{\theta}_k^+ = \bar{x}_k - x_k$$  \hspace{1cm} (2-33)$$

$$\hat{\theta}_{k+1}^- = \bar{x}_{k+1} - x_{k+1}$$  \hspace{1cm} (2-34)$$

The correction step is a linear combination of the propagated states, inputs and measurements.

$$\bar{x}_{k+1}^+ = K_{k+1} \hat{\theta}_{k+1}^- + \bar{x}_{k+1}$$  \hspace{1cm} (2-35)$$

Hence the error in the corrected estimate will be,

$$\hat{\theta}_k^+ = \bar{x}_k^+ - x_k$$  \hspace{1cm} (2-36)$$

Subtracting \(x_{k+1}\) from both sides of equation (2-35) and using linear approximation for \(z_{k+1}\) from equation (2-32),

$$\hat{\theta}_{k+1}^+ = K_{k+1} \bar{x}_{k+1}^- - x_{k+1} + \bar{K}_{k+1} u_{meas,k+1} + K_{k+1} (C_{k+1} x_{k+1} + D_{k+1} u_{meas,k+1} + E_{k+1} w_{k+1} + I v_{k+1})$$

$$= K_{k+1} (\bar{x}_{k+1}^- - x_{k+1}) + (K_{k+1} + C_{k+1} K_{k+1} - I) x_{k+1} + \bar{K}_{k+1} + K_{k+1} (E_{k+1} w_{k+1} + v_{k+1})$$

$$= K_{k+1} (\bar{x}_{k+1}^- + (K_{k+1} + C_{k+1} K_{k+1} - I) x_{k+1} + (K_{k+1} E_{k+1} w_{k+1} + \bar{K}_{k+1} + K_{k+1} (E_{k+1} w_{k+1} + v_{k+1}))$$  \hspace{1cm} (2-37)$$
2 Theory

For achieving unbiased estimates by the Kalman filter, the mean of the above estimation error should be zero, that is \( E[\hat{e}^+_{k+1}] = 0 \). Therefore taking expectation on both sides of equation (2-37)

\[
E[\hat{e}^+_{k+1}] = K_{k+1}^* E[\hat{e}^-_{k+1}] + (K_{k+1}^* + K_{k+1}C_{k+1} - I)E[x_{k+1}]
+ (K_{k+1}D_{k+1} + \bar{K}_{k+1})E[u_{meas,k+1}] + K_{k+1}(E[w_{k+1}] + E[v_{k+1}])
\]

(2-38)

To obtain the desired result in the above equation, the contribution of the states and inputs along its right hand side should be made zero; and for this the following expression should hold, keeping in mind \( E[w_{k+1}] = 0 \) and \( E[v_{k+1}] = 0 \). That is,

\[
\begin{align*}
K_{k+1}^* &= (I - K_{k+1}C_{k+1}) \\
\bar{K}_{k+1} &= -K_{k+1}D_{k+1}
\end{align*}
\]

(2-39)

And it leads to the linear combination,

\[
\hat{x}^+_{k+1} = (I - K_{k+1}C_{k+1})\hat{x}^-_{k+1} + K_{k+1}z_{k+1} - K_{k+1}D_{k+1}u_{meas,k+1}
= \hat{x}^-_{k+1} + K_{k+1}\left( z_{k+1} - D_{k+1}u_{meas,k+1} - C_{k+1}\hat{x}^-_{k+1} \right)
\]

(2-40)

Again in the above equation, if the feed-through matrix \( D_{k+1} \) is replaced with zero, the standard expression for the corrected output can be obtained, as provided in equation (2-63).

If the true nonlinear output is used instead and assuming noise to be zero,

\[
\hat{x}^+_{k+1} = \hat{x}^-_{k+1} + K_{k+1}\left( z_{k+1} - h(\hat{x}^-_{k+1}, u_{meas,k+1}, 0, 0) \right)
\]

(2-41)

This expression is not used everywhere, but since for the present case the output equations are known, hence it is better to use the true nonlinear output which is always more accurate than the linearized form, due to the presence of inherent linearization error in the latter case.

The matrix \( K_{k+1} \) is called as the Kalman Gain matrix and if its value is known at this point, the above equation would result in a simple way to correct the predicted estimates \( \hat{x}^-_{k+1} \) using measurements \( z_{k+1} \). \( K_{k+1} \) in a literal sense is a weighing measure for the corrected states \( \hat{x}^+_{k+1} \), that weighs the influence of the measurements over the ‘prediction step’ (or propagated states). That is, it weights the contribution of the difference between the measured and propagated outputs on one side, with the contribution of the predicted states on the other side, in obtaining the estimates for the corrected states.

Also the corresponding error from equation (2-37) using equation (2-39) is,

\[
\hat{e}^+_{k+1} = \hat{x}^+_{k+1} - x_{k+1}
= (I - K_{k+1}C_{k+1})\hat{e}^-_{k+1} + K_{k+1}E_{k+1}w_{k+1} + K_{k+1}v_{k+1}
\]

(2-42)

**Propagation of states and covariance matrix**

After all the linearization, the first step prediction is done by a nonlinear propagation of the states without the stochastic part, which forms the basis of the prediction step i.e., to estimate the states based upon the knowledge of the system. This can be done using Euler’s method or third, fourth or fifth order Runge-Kutta (R-K) method. The fourth order R-K method evaluates one extra function than third order, hence has lower value of Global truncation error (GTE).
Even though fifth order has lower GTE than fourth order, the former is computationally more demanding and also the difference in GTE’s is not too much. Therefore fourth order R-K method was used (details of which can be seen in appendix A.3).

\[
\bar{x}_{k+1} = \bar{x}_k + \int_{t_k}^{t_{k+1}} f(x(\tau), u_{meas}(\tau), 0) d\tau
\]  

(2-43)

And, its linear counterpart would be,

\[
\bar{x}_{k+1} = \Phi_k \bar{x}_k + \Psi_k \mathbf{B}_k \mathbf{u}_{meas,k}
\]  

(2-44)

On the other hand, the state covariance matrix is propagated in a linear way.

\[
P_{k+1} = E[(\bar{x}_{k+1} - x_{k+1})(\bar{x}_{k+1} - x_{k+1})^T]
\]  

(2-45)

Substituting equation (2-31) and (2-44) in (2-45)

\[
P_{k+1} = E \left[ \left( \Phi_k \hat{e}_k^+ - \Psi_k \mathbf{F}_k \mathbf{w}_k \right) \left( \Phi_k \hat{e}_k^+ - \Psi_k \left( \mathbf{F}_k \mathbf{w}_k \right) \right)^T \right]
\]

\[
= \Phi_k E \left[ \hat{e}_k^+ \hat{e}_k^{+T} \right] \Phi_k^T + \Psi_k \mathbf{F}_k \cdot E \left[ \mathbf{w}_k \mathbf{w}_k^T \right] \mathbf{F}_k^T \Psi_k^T - \Phi_k E \left[ \hat{e}_k^+ \mathbf{w}_k^T \right] \mathbf{F}_k^T \Psi_k^T
\]

(2-46)

\[
- \Psi_k \mathbf{F}_k E \left[ \mathbf{w}_k \hat{e}_k^{+T} \right] \Phi_k^T
\]

All the processes are assumed to be stationary, that is no correlation in time between state error \( \hat{e}_k^+ \), process, measurement and input noise is present. This leads to \( E[\hat{e}_k^+ \mathbf{w}_k^T] = 0 \) and \( E[\mathbf{v}_k \mathbf{w}_k^T] = 0 \).

\[
E \left[ \hat{e}_k^+ \mathbf{w}_k^T \right] = E \left[ ((I - \mathbf{K}_k \mathbf{C}_k) \hat{e}_k^+ + \mathbf{K}_k \mathbf{E}_k \mathbf{w}_k + \mathbf{K}_k \mathbf{v}_k) \mathbf{w}_k^T \right]
\]

\[
= \mathbf{K}_k \mathbf{E}_k E \left[ \mathbf{w}_k \mathbf{w}_k^T \right]
\]  

(2-47)

The propagation of the state covariance matrix thus becomes

\[
P_{k+1} = \Phi_k E \left[ \hat{e}_k^+ \hat{e}_k^{+T} \right] \Phi_k^T + \Psi_k \mathbf{F}_k \cdot E \left[ \mathbf{w}_k \mathbf{w}_k^T \right] \mathbf{F}_k^T \Psi_k^T - \Phi_k \mathbf{K}_k \mathbf{E}_k E \left[ \mathbf{w}_k \mathbf{w}_k^T \right] \mathbf{F}_k^T \Psi_k^T
\]

\[
- \Psi_k \mathbf{F}_k E \left[ \mathbf{w}_k \hat{e}_k^{+T} \right] \Phi_k^T
\]  

(2-48)

Using the definition of covariance matrix,

\[
P_{k+1} = \Phi_k P_k^+ \Phi_k^T + \Psi_k \mathbf{F}_k E \left[ \mathbf{w}_k \mathbf{w}_k^T \right] \mathbf{F}_k^T \Psi_k^T - \Phi_k \mathbf{K}_k \mathbf{E}_k E \left[ \mathbf{w}_k \mathbf{w}_k^T \right] \mathbf{F}_k^T \Psi_k^T
\]

\[
- \Psi_k \mathbf{F}_k E \left[ \mathbf{w}_k \hat{e}_k^{+T} \right] \Phi_k^T
\]  

(2-49)

One can note at this point that, by substituting the feed-through matrix \( \mathbf{E}_k \) to be zero in the above equation, the corresponding expression for the standard EKF formulation, as given in equation (2-60), can be obtained.
Correction update using measurements

The expression for updated states is the same as given in equation (2-41) and is mentioned here as well.

\[
\hat{x}_{k+1} = \tilde{x}_{k+1} + K_{k+1} \left( z_{k+1} - h(\tilde{x}_{k+1}, u_{\text{meas},k+1}, 0, 0) \right) \quad (2-50)
\]

Based on above theory, an expression for the updated covariance matrix can be written as

\[
P_{k+1}^+ = E \left[ \hat{e}_{k+1}^+ \hat{e}_{k+1}^{+T} \right] \quad (2-51)
\]

Using error equation (2-42)

\[
P_{k+1}^+ = \left( I - K_{k+1} C_{k+1} \right) P_{k+1}^- \left( I - K_{k+1} C_{k+1} \right)^T + K_{k+1} E \left[ w_{k+1} w_{k+1}^T \right] E_{k+1}^T + E \left[ v_{k+1} v_{k+1}^T \right] K_{k+1}^T
\]

\[
+ \left( I - K_{k+1} C_{k+1} \right) E \left[ \hat{e}_{k+1}^+ w_{k+1}^T \right] E_{k+1}^T K_{k+1}^T
\]

\[
+ \left( I - K_{k+1} C_{k+1} \right) E \left[ \hat{e}_{k+1}^+ v_{k+1}^T \right] K_{k+1}^T
\]

\[
+ K_{k+1} E_{k+1} \left( I_{k+1} E \left[ w_{k+1} v_{k+1}^T \right] K_{k+1}^T \right)
\]

\[
P_{k+1}^+ = \left( I - K_{k+1} C_{k+1} \right) P_{k+1}^- \left( I - K_{k+1} C_{k+1} \right)^T + K_{k+1} E \left[ w_{k+1} v_{k+1}^T \right] K_{k+1}^T
\]

\[
+ K_{k+1} E_{k+1} \left( I_{k+1} E \left[ w_{k+1} v_{k+1}^T \right] K_{k+1}^T \right)
\]

where the expression for updated states is the same as given in equation (2-51) and setting it to zero. That is,

\[
\frac{\partial}{\partial K_{k+1}} \text{tr}(P_{k+1}^+) \mid_{P_{k+1}^+ = 0} = 0
\]

(2-54)

This results in the optimal \( K_{k+1} \) which will yield an efficient state estimator. An expression from matrix calculus is needed to be able to compute equation (2-54)

\[
\frac{\partial}{\partial A} \left( \text{tr}(ABA^T) \right) = 2AB
\]

(2-55)
Using the result of equation (2-55) in equation (2-54)

\[
\frac{\partial}{\partial K_{k+1}} \text{tr}(P_{k+1}^+) = 2 \cdot (I - K_{k+1} C_{k+1}) P_{k+1}^+ (-C_{k+1}^T) + 2 \cdot K_{k+1} (E_{k+1} E[w_{k+1} w_{k+1}^T] E_{k+1}^T + E[v_{k+1} v_{k+1}^T])^{-1} \frac{1}{2} \tag{2-56}
\]

\[
K_{k+1} = P_{k+1}^- C_{k+1}^T (C_{k+1} P_{k+1}^- C_{k+1}^T + E_{k+1} E[w_{k+1} w_{k+1}^T] E_{k+1}^T + E[v_{k+1} v_{k+1}^T])^{-1} \tag{2-57}
\]

Two limiting cases for above expression are worth noting. For very large values of the propagation error covariance \( P_{k+1}^- \), compared to \( E[w_{k+1} w_{k+1}^T] \) and \( E[v_{k+1} v_{k+1}^T] \), that is considering very large uncertainties in prediction step, the influence of term \( E[v_{k+1} v_{k+1}^T] \) in calculating \( K_{k+1} \) will be negligible and will result in something like a weighted pseudo-inverse of \( C_{k+1} \). Using this result in equation (2-50) one can realize that the corrected states \( \hat{x}_{k+1}^+ \) will be dominated by the measurements \( x_{k+1} \).

On the other hand, for large values of the measurement uncertainties \( E[v_{k+1} v_{k+1}^T] \) compared to \( P_{k+1}^- \) (the contribution of \( E[w_{k+1} w_{k+1}^T] \) is small here as the magnitude of the feed-through matrix \( E_{k+1} \) is usually very small), the Kalman gain matrix \( K_{k+1} \) will be negligibly small and thus the corrected states \( \hat{x}_{k+1}^+ \) in equation (2-50) will be dominated by predicted states \( \hat{x}_{k+1}^- \) only.

By using the result of equation (2-57) in equation (2-53) and after some lengthy computations, a simpler form of the correction equation for state covariance matrix \( P_{k+1}^+ \) can be obtained as,

\[
P_{k+1}^+ = (I - K_{k+1} C_{k+1}) P_{k+1}^-
\]

Even though the above form is easier to implement, but seldom used for numerical solutions as the resulting update might make the covariance matrix non-symmetric because of round-off errors and finite machine precision. But nevertheless, the obtained final equation, for the corrected covariance matrix, is observed to be the same as presented in equation (2-62).

**Algorithm of EKF for a standard system (without feed-through)**

A. **Prediction step:** propagation of the states and error covariance matrix at time instant \( k \).

\[
\hat{x}_{k+1}^- = \hat{x}_k^+ + \int_{x_k}^{x_{k+1}} f(x(\tau), u_{meas}(\tau), 0) d\tau \tag{2-59}
\]

\[
P_{k+1}^- = \Phi_k P_k^+ \Phi_k^T + \Psi_k F_k E[w_k w_k^T] F_k^T \Psi_k^T
\]

\begin{align*}
&= \Phi_k P_k^+ \Phi_k^T + \Psi_k F_k Q_k F_k^T \Psi_k^T \\
&= \Phi_k P_k^+ \Phi_k^T + \Psi_k F_k Q_k F_k^T \Psi_k^T
\end{align*}

Where, matrices \( F_k, \Phi_k \) and \( \Psi_k \) are computed is the same way as in equations (2-21), (2-22) and (2-23) respectively.
2.3.2 Extended Rauch-Tung-Striebel Smoother

Since the data is processed offline here, no real-time requirements have to be met. This leads to the advantage of forward and backward processing of data in time and forms the basis for fixed interval smoothing. A two filter formulation, one running forward in time and other running backward, may be employed. The smoothed estimate then can be expressed as a linear combination of the forward and the backward estimate.

Data is first processed in a forward pass starting from \( k = 0 \) up to \( k = N \) using extended kalman filter, considering measurements from \( 0, \ldots, k \). Then the same data set is processed in a backward pass from \( k = N \) to \( k = 0 \), considering measurements from \( N, \ldots, k \). The final estimates are known as smoothed estimates which are the linear combination of the two and are evaluated at every point in time. (Notation: superscripts \( f, b, s \) indicate forward, backward and smoothed values)

\[
\hat{x}^s_k = M^f_k \hat{x}^f_k + M^b_k \hat{x}^b_k
\]

Given that,

\[
M^f_k + M^b_k = I
\]

Since in both the pass, two different sets of measurements are processed, hence the two estimates can be regarded as uncorrelated. The smoothed state error covariance matrix becomes

\[
P_k^s = Cov[x_k^s] = Cov[M^f_k \hat{x}^f_k + (1 - M^f_k) \hat{x}^b_k]
\]

\[
= M^f_k Cov[\hat{x}^f_k] (M^f_k)^T + (1 - M^f_k) Cov[\hat{x}^b_k] (1 - M^f_k)^T
\]

\[
= M^f_k P_k^f (M^f_k)^T + (1 - M^f_k) P_k^b (1 - M^f_k)^T
\]
As already seen in the derivation of Kalman filter, minimum variance estimator is desired, that is \( M_k^f \) is chosen such that the trace of \( P_k^f \) is minimised.

\[
\frac{\partial}{\partial M_k^f} \text{tr}(P_k^f) = \frac{\partial}{\partial M_k^f} \text{tr} \left( M_k^f P_k^{f+} (M_k^f)^T + (1 - M_k^f) P_k^{b+} (1 - M_k^f)^T \right)
\]  
\[
\text{(2-67)}
\]

Using the result of equation (2-55),

\[
\frac{\partial}{\partial M_k^f} \text{tr}(P_k^f) = 2 \cdot M_k^f P_k^{f+} - 2 \cdot (1 - M_k^f) P_k^{b+} = 0
\]
\[
\text{(2-68)}
\]

\[
M_k^f = P_k^{b+} (P_k^{b+} + P_k^{f+})^{-1}
\]
\[
\text{(2-69)}
\]

\[
M_k^b = P_k^{f+} (P_k^{b+} + P_k^{f+})^{-1}
\]
\[
\text{(2-70)}
\]

After some mathematical computations, two rather simple equations for smoothed state vector \( x_k^s \) and smoothed covariance matrix \( P_k^s \) are obtained. Again (Simon, 2006) can be consulted for a detailed derivation.

\[
P_k^s = \left( (P_k^{b+})^{-1} + (P_k^{f+})^{-1} \right)^{-1}
\]
\[
\text{(2-71)}
\]

\[
x_k^s = P_k^s \left( (P_k^{f+})^{-1} \hat{x}_k^f + (P_k^{b+})^{-1} \hat{x}_k^{b+} \right)
\]
\[
\text{(2-72)}
\]

By intuition, it can be seen that if the estimates obtained during forward pass are efficient than the ones obtained during backward pass, that is, if forward state covariance \( P_k^{f+} \) is smaller than the backward state covariance \( P_k^{b+} \), the resulting weight \( M_k^f \) will be rather big and will result in higher weightage of \( \hat{x}_k^{f+} \) in calculation of smoothed estimates.

Till now, three steps are followed: a forward pass through the filter, a backward pass and then the smoothing step. But this type of two filter formulation is computationally more demanding and to overcome this problem, Rauch, Tung and Striebel introduced the RTS smoother, which combines the backward filter and the smoothing step into a single-step correction to the forward filter estimate. That is, the backward pass and smoothing step become one in RTS smoother. The derivation is lengthy and will be omitted here but for details one can refer (Simon, 2006). The resulting equations for smoothed states, covariance matrix and weights are,

\[
M_k = P_k^{f+} \Phi_k^T (P_{k+1}^{f-})^{-1}
\]
\[
\text{(2-73)}
\]

\[
P_k^s = P_k^{f+} + M_k (P_{k+1}^s - P_{k+1}^{f-}) M_k^T
\]
\[
\text{(2-74)}
\]

\[
x_k^s = \hat{x}_k^{f+} + M_k (x_{k+1}^s - \hat{x}_{k+1}^{f-})
\]
\[
\text{(2-75)}
\]

These quantities are equal for linear and extended filtering but the only difference is that for nonlinear case \( \Phi_k^T \) is time dependent as can be seen from equation (2-22).
The Extended RTS can be summarized as,

A. Initialize the EKF

\[
\hat{x}_0^{f+} = E[x_0] \\
P_0^{f+} = E \left[ \left( x_0 - \hat{x}_0^{f+} \right) \left( x_0 - \hat{x}_0^{f+} \right)^T \right] \tag{2-76}
\]

B. Compute the forward pass and save the quantities: \( \hat{x}_k^{f+}, \hat{x}_k^{f-}, P_k^{f+}, P_k^{f-}, \Phi_k \).

C. Initialize Extended RTS smoother as

\[
x_N^s = \hat{x}_N^{f+} \\
P_N^s = P_N^{f+} \tag{2-77}
\]

A. Run the combined backward pass and smoothing correction for \( k = N - 1, ..., 1 \)

\[
M_k = P_k^{f+} \Phi_k^T (P_{k+1}^{f+})^{-1} \tag{2-78}
\]

\[
P_k^{s+} = P_k^{f+} + M_k (P_{k+1}^{s+} - P_{k+1}^{f-})M_k^T \tag{2-79}
\]

\[
x_k^s = \hat{x}_k^{f+} + M_k (x_{k+1}^s - \hat{x}_{k+1}^{f-}) \tag{2-80}
\]

2.3.3 Problems with EKF

Even though EKF is most widely used filtering strategy for the nonlinear systems, it can still be difficult to implement with tuned parameters and often gives unreliable estimates if the system under observation has severe nonlinearities, for instance, the nonlinearities arise during ‘agile manoeuvres of a fighter aircraft’. That is, EKF is more reliable for the systems which are piecewise linear within the update interval. This is on account of the dependence of EKF algorithm, on linearization (of the states and outputs about the current estimate) to propagate the mean and covariance of the state as stated in (Simon, 2006), and in some cases this may also result in the divergence of the filter.

Also according to (Simon, 2006), “Linearization approximations often result in errors in the transformation of mean and covariance when a random variable is operated on, by a nonlinear function”. These errors arise due to the truncated terms (all the terms higher than first order) of the infinite series that are neglected in the linear approximation. Other aspects of Linearization are:

- It can produce highly unstable filtering performance if the time step of propagation is not sufficiently small.
- Furthermore, sufficiently small time step usually leads to high computational burden, as the number of computations required for the calculation of the Jacobians as well as the estimated states and covariance matrices, are large.

Furthermore, the linearization can only be applied if the Jacobian matrices exist; as sometimes the nonlinear function is not differentiable, for example, in case of the systems with discontinuities and thus limiting the application of EKF.
2.4 Unscented Transformation

The problem that often arises with a nonlinear system is that, it is difficult to transform a probability density function (pdf) through a general nonlinear function. The EKF works on the principle that “a linearized transformation of mean and covariance is approximately equal to the true nonlinear transformations” as mentioned in (Simon, 2006), but the approximations could be sometimes unsatisfactory.

For the derivation of equations a state vector \( x \in \mathbb{R}^L \) is considered, where \( L \) are the number of states and \( f(x) \) is a vector function of the nonlinear dynamic system, which is represented as

\[
y = f(x)
\]  

Using the Taylor series, the nonlinear system can be approximated around the mean \( \bar{x} \) as

\[
y = f(\bar{x}) + D_x f + \frac{1}{2!} D_x^2 f + \frac{1}{3!} D_x^3 f + \ldots
\]  

Where, \( D_x f \) is a differential operator operating on the nonlinear function \( f(x) \) and is defined as

\[
D_x^k f = \left( \sum_{i=1}^{L} \frac{\partial f(x)}{\partial x_i} \right)^k f(x) \bigg|_{x=\bar{x}}
\]  

Where, \( \bar{x} = x - \bar{x} \) and \( x_i \) represents the \( i^{th} \) state of the state vector \( x \).

For example if \( k = 1 \),

\[
D_x^1 f = \sum_{i=1}^{L} \left. \frac{\partial f(x)}{\partial x_i} \right|_{x=\bar{x}} = \left( \frac{\partial f(x)}{\partial x_1} + \frac{\partial f(x)}{\partial x_2} + \frac{\partial f(x)}{\partial x_3} + \ldots \right) f(x) \bigg|_{x=\bar{x}}
\]  

Also if \( k = 2 \),

\[
D_x^2 f = \left( \sum_{i=1}^{L} \frac{\partial f(x)}{\partial x_i} \right)^2 f(x) \bigg|_{x=\bar{x}} = \sum_{i,j=1}^{L} \left. \left( \frac{\partial f(x)}{\partial x_i} \right) \left( \frac{\partial f(x)}{\partial x_j} \right) f(x) \right|_{x=\bar{x}}
\]

\[
= \left( \frac{\partial f(x)}{\partial x_1} + \frac{\partial f(x)}{\partial x_2} + \frac{\partial f(x)}{\partial x_3} + \ldots \right) \left( \frac{\partial f(x)}{\partial x_1} + \frac{\partial f(x)}{\partial x_2} + \frac{\partial f(x)}{\partial x_3} + \ldots \right) f(x) \bigg|_{x=\bar{x}}
\]

Where, \( \frac{\partial}{\partial x_i} \) is a column vector.

Now by taking the expectation on both sides of equation (2-82), mean of \( y \) can be expanded as

\[
\bar{y} = E \left[ f(\bar{x}) + D_x f + \frac{1}{2!} D_x^2 f + \frac{1}{3!} D_x^3 f + \ldots \right]
\]
\[ \bar{y} = f(\bar{x}) + E\left[ D_{\bar{x}} f + \frac{1}{2!} D_{\bar{x}}^2 f + \frac{1}{3!} D_{\bar{x}}^3 f + \ldots \right] \]  

(2-87)

Evaluating the terms separately,

\[
E[D_{\bar{x}} f] = E\left[ \sum_{i=1}^{L} \bar{x}_i \frac{\partial f(x)}{\partial x_i} \bigg|_{x=\bar{x}} \right] \\
= \sum_{i=1}^{L} E[\bar{x}_i] \frac{\partial f(x)}{\partial x_i} \bigg|_{x=\bar{x}}
\]

(2-88)

To obtain the value of above expression, a simple case is considered

\[
\because E(\bar{x}_i) = E(x_i - \bar{x}_i) \\
= E(x_i) - E(\bar{x}_i) \\
= \bar{x}_i - \bar{x}_i = 0
\]

(2-89)

A detailed proof to show that, all odd order moments of a zero-mean R.V (Random variable) with a symmetric probability density function (pdf) are zero, can be seen in appendix A.4.

Therefore, assuming \( x \) to be a symmetrically distributed R.V. about the mean (\( \bar{x} \)) and using the result obtained in equation (2-89)

\[
E[D_{\bar{x}} f] = 0
\]

(2-90)

Similarly,

\[
E[D_{\bar{x}}^3 f] = E\left[ \left( \sum_{i=1}^{L} \bar{x}_i \frac{\partial f(x)}{\partial x_i} \right)^3 \bigg|_{x=\bar{x}} \right] \\
= \sum_{i,j,k=1}^{L} E[\bar{x}_i \bar{x}_j \bar{x}_k] \left\{ \left( \frac{\partial}{\partial \bar{x}_i} \right) \left( \frac{\partial}{\partial \bar{x}_j} \right) \left( \frac{\partial}{\partial \bar{x}_k} \right) f(\bar{x}) \right|_{x=\bar{x}}
\]

(2-91)

As can be seen in above equation, the sum consist only of third order moments which are of the form, \( E[\bar{x}_i^3] \), \( E[\bar{x}_i^2 \bar{x}_j] \) and \( E[\bar{x}_i \bar{x}_j \bar{x}_k] \); and based on the result of appendix A.4, its expected value will always be zero. Therefore, substituting \( E[\bar{x}_i \bar{x}_j \bar{x}_k] = 0 \) in equation (2-91), the following result can be obtained.

\[
E[D_{\bar{x}}^3 f] = 0
\]

(2-92)
Likewise all the odd terms in equation (2-87) would be zero. This leads to a further simplification of the equation (2-87),

\[
\bar{y} = f(\bar{x}) + \frac{1}{2!} E[D^2_\bar{x}f] + \frac{1}{4!} E[D^4_\bar{x}f] + ... \tag{2-93}
\]

Now, using the definition of equation (2-85)

\[
E[D^2_\bar{x}f] = E \left[ \sum_{i,j=1}^{L} \left( \frac{\partial}{\partial x_i} \bar{x}_i \frac{\partial}{\partial x_j} \bar{x}_j \right) f(x) \bigg| _{x = \bar{x}} \right]
\]

\[
E[D^2_\bar{x}f] = E \left[ \sum_{i,j=1}^{L} \left( \bar{x}_i \frac{\partial}{\partial x_i} \bar{x}_j \frac{\partial}{\partial x_j} \right) f(x) \bigg| _{x = \bar{x}} \right]
\]

\[
= \left( \sum_{i,j=1}^{L} \frac{\partial}{\partial x_i} E[\bar{x}_i \bar{x}_j] \frac{\partial}{\partial x_j} \right) f(x) \bigg| _{x = \bar{x}} \tag{2-94}
\]

\[
= \left( \sum_{i,j=1}^{L} \Delta_i E[\bar{x}_i \bar{x}_j] \Delta_j \right) f(x) \bigg| _{x = \bar{x}}
\]

\[
= \left( \sum_{i,j=1}^{L} \Delta_i P_{ij} \Delta_j \right) f(x) \bigg| _{x = \bar{x}}
\]

Where, \( \Delta_i = \frac{\partial}{\partial x_i} \) is the \( i^{th} \) column of the row vector \( \Delta \) and \( P_{ij} \) is the element in the \( i^{th} \) row and \( j^{th} \) column of the covariance matrix \( P_x \), which are defined as

\[
\Delta = \begin{bmatrix} \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_2} & \frac{\partial}{\partial x_3} & \ldots \end{bmatrix}
\]

\[
P_x = E[\bar{x} \cdot \bar{x}^T] \tag{2-95}
\]

Therefore, using the definition in equation (2-95), equation (2-94) can be written in the vector form as,

\[
E[D^2_\bar{x}f] = (\Delta P_x \Delta^T) f(x) \bigg| _{x = \bar{x}} \tag{2-96}
\]

By substituting the above result in equation (2-93), an analytic expression for the mean of the Taylor series expansion of the nonlinear system, can be obtained

\[
\bar{y} = f(\bar{x}) + \frac{1}{2} (\Delta P_x \Delta^T) f(x) \bigg| _{x = \bar{x}} + \frac{1}{4!} E[D^4_\bar{x}f] + ... \tag{2-97}
\]

**Unscented Transform**

According to (Hayking, 2001) “The unscented transformation (UT) is a method for calculating the statistics of a random variable which undergoes a nonlinear transformation”. A random variable \( x \in \mathbb{R}^L \) is propagated through a nonlinear vector function, \( y = f(x) \) with the assumption that, \( x \) has a mean equal to \( \bar{x} \) and a covariance matrix given by \( P_x \). A schematic
To be able to calculate the statistics of $y$, a matrix $\chi$ is formed with $2L + 1$ sigma vectors according to the following equation.

$$
\begin{align*}
\chi_0 &= \bar{x} \\
\chi_i &= \bar{x} + \left(\sqrt{(L + \lambda)}P_x\right)_i & i = 1, \ldots, L \\
\chi_i &= \bar{x} - \left(\sqrt{(L + \lambda)}P_x\right)_{i-L} & i = L + 1, \ldots, 2L
\end{align*}
$$

(2-98)

Where, $\chi_i \in \mathbb{R}^L$ is the $i^{th}$ sigma vector of the matrix $\chi$ and $\left(\sqrt{(L + \lambda)}P_x\right)_i$ represents the $i^{th}$ column of a matrix square root of the covariance matrix $P_x$. This matrix square root can be computed using ‘Cholesky’ or ‘Schur’ decomposition algorithm, details of which can be seen in (Simon, 2006).

These generated sigma vectors are then propagated through the nonlinear function yielding a cloud of transformed points,

$$
Y_i = f(\chi_i) \quad i = 0, \ldots, 2L
$$

(2-99)

And the estimated mean and covariance of the predicted state are then approximated, using a weighted sample mean and weighted sample covariance of the posterior sigma points.

$$
\bar{y} \approx \sum_{i=0}^{2L} W_i^{(m)} Y_i
$$

(2-100)

$$
P_y = \sum_{i=0}^{2L} W_i^{(c)} (Y_i - \bar{y})(Y_i - \bar{y})^T
$$

(2-101)

Where the weights $W_i$ are evaluated as,
\[ W_0^{(m)} = \frac{\lambda}{L + \lambda} \]

\[ W_0^{(c)} = \frac{\lambda}{L + \lambda} + 1 - \alpha^2 + \beta \]  

\[ W_i^{(m)} = W_i^{(c)} = \frac{1}{2(L + \lambda)} \quad i = 1, ..., 2L \]  

Where, \( \lambda = \alpha^2(L + \kappa) - L \)

\( \alpha = \) Spread of the sigma points around \( \bar{x} \). It is usually set to a small positive value as a larger value would set sigma points far away from the mean.

\[ \approx (10^{-4} \leq \alpha < 1) \]

\( \kappa = \) Secondary scaling parameter and is set equal to \((3 - L)\), based on a heuristic approach.

\( \beta = \) Incorporate a prior knowledge about the distribution

\[ \approx 2 \text{ (Optimal for Gaussian distribution)} \]

The values of the above parameters do not have much influence on the unscented transformations but they do affect the accuracy as can be seen later that the scaling parameters like \( \lambda \) and \( \kappa \), reduces the error in evaluating the fourth and higher order sample moments of the sigma points.

Also, \( \alpha \) (spread of the sigma points) affects the primary scaling parameter \( \lambda \), which in turn affects the evaluations of the corresponding weights. That is, if \( \alpha = 10^{-4} \) (the sigma points are very near to the mean), the magnitude of \( W_0^{(m)} \) and \( W_0^{(c)} \) will the higher than \( W_i^{(m)} \) (or \( W_i^{(c)} \)), and thus the initial sigma points, which are near to the mean, would be highly weighted in computing the new mean and covariance matrix. On the other hand, if \( \alpha = 1 \), the magnitude of \( W_i^{(m)} \) (or \( W_i^{(c)} \)) will the higher than \( W_0^{(m)} \) and \( W_0^{(c)} \) and therefore the farther sigma points will have a higher weightage in the mean and covariance evaluation.

**Accuracy of the mean**

Based on the discussed unscented transformations, firstly an expression for the posterior mean of the unscented transformation will be evaluated and compared with the true expression of the mean already got in equation \((2-97)\). And later, the accuracy of the mean by UT and by linearization (EKF approach) will be compared to obtain a better method.

The UT calculates the posterior mean by first generating the sigma points about the known mean, using the equation \((2-98)\) and representing \( \sigma_i \) to be the \( i^{th} \) column of the matrix square root of \( P_x \). That is

\[ x_i = \bar{x} \pm \sqrt{(L + \lambda)}\sigma_i \]

\[ = \bar{x} \pm \sigma_i \]  

\[ (2-103) \]
Where,

\[
P_x = \sum_{i,j=1}^{L} (\sigma_i \sigma_j^T) = \sum_{i,j=1}^{L} (\sigma_i^T \sigma_j^T) \quad \bar{\sigma}_i = \sqrt{(L + \lambda)} \sigma_i
\]  

(2-104)

Using the above formulation of sigma points, the propagation of each point through the nonlinear function can be rewritten in the form of a Taylor series expansion about the mean \( \bar{x} \), as

\[
Y_i = f(x_i) = f(\bar{x}) + D_{\bar{x}}f + \frac{1}{2!} D_{\bar{x}}^2 f + \frac{1}{3!} D_{\bar{x}}^3 f + \frac{1}{4!} D_{\bar{x}}^4 f + \ldots
\]  

(2-105)

Where, the definition of \( D_{\bar{x}}f \) is given below, similar to equation (2-83)

\[
D_{\bar{x}}f = \left( \sum_{k=1}^{L} \bar{\sigma}_{i,k} \frac{\partial}{\partial x_k} \right) f(x) \bigg|_{x = \bar{x}}
\]  

(2-106)

Here, \( \bar{\sigma}_{i,k} \) represents the \( k \)th element of the \( i \)th column vector \( \bar{\sigma}_i \).

From equation (2-100),

\[
\bar{y}_{ut} = \sum_{i=0}^{2L} W_i^{(m)} Y_i
\]

\[
= \frac{\lambda}{L + \lambda} f(\bar{x}) + \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \left( f(\bar{x}) + D_{\bar{x}}f + \frac{1}{2!} D_{\bar{x}}^2 f + \frac{1}{3!} D_{\bar{x}}^3 f + \frac{1}{4!} D_{\bar{x}}^4 f + \ldots \right)
\]

\[
= f(\bar{x}) + \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \left( D_{\bar{x}}f + \frac{1}{2!} D_{\bar{x}}^2 f + \frac{1}{3!} D_{\bar{x}}^3 f + \frac{1}{4!} D_{\bar{x}}^4 f + \ldots \right)
\]  

(2-107)

Using the same reason as used before, that is, as the sigma points are symmetrically distributed around \( \bar{x} \), all the odd moments are zero, a simplified equation can be obtained,

\[
\bar{y}_{ut} = f(\bar{x}) + \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \left( \frac{1}{2!} D_{\bar{x}}^2 f + \frac{1}{4!} D_{\bar{x}}^4 f + \frac{1}{6!} D_{\bar{x}}^6 f + \ldots \right)
\]  

(2-108)

Considering the second term of the above equation and using the expression given in equation (2-106),
\[
\frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \frac{1}{2!} D_{\bar{x}}^{2} f = \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \frac{1}{2!} \left( \sum_{k=1}^{L} \frac{\partial}{\partial x_k} \right) \left( \sum_{m=1}^{L} \frac{\partial}{\partial x_m} \right) f(x) \bigg|_{x = \bar{x}}
\]

\[
= \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \frac{1}{2!} \left( \sum_{k=1}^{L} \frac{\partial}{\partial x_k} \left( \sum_{m=1}^{L} \frac{\partial}{\partial x_m} \right) \right) f(x) \bigg|_{x = \bar{x}}
\]

\[
= \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \frac{1}{2!} \left( \sum_{i,j=1}^{L} \frac{\partial}{\partial x_i} \left( \sum_{k=1}^{L} \frac{\partial}{\partial x_k} \right) \right) f(x) \bigg|_{x = \bar{x}}
\]

\[
\frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \frac{1}{2!} D_{\bar{x}}^{2} f = \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \frac{1}{2!} \left[ \sum_{i,j=1}^{L} \frac{\partial}{\partial x_i} \left( \sum_{k=1}^{L} \frac{\partial}{\partial x_k} \right) \right] f(x) \bigg|_{x = \bar{x}}
\]

\[
= \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \frac{1}{2!} \left[ \sum_{i,j=1}^{L} \frac{\partial}{\partial x_i} \left( \sum_{k=1}^{L} \frac{\partial}{\partial x_k} \right) \right] f(x) \bigg|_{x = \bar{x}}
\]

Therefore, in the vector form

\[
\frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \frac{1}{2!} D_{\bar{x}}^{2} f = \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \frac{1}{2!} \left( \sum_{i,j=1}^{L} \frac{\partial}{\partial x_i} \left( \sum_{k=1}^{L} \frac{\partial}{\partial x_k} \right) \right) f(x) \bigg|_{x = \bar{x}}
\]

\[
= \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \frac{1}{2!} \left( \sum_{i,j=1}^{L} \frac{\partial}{\partial x_i} \left( \sum_{k=1}^{L} \frac{\partial}{\partial x_k} \right) \right) f(x) \bigg|_{x = \bar{x}}
\]

\[
= \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \frac{1}{2!} \left( \sum_{i,j=1}^{L} \frac{\partial}{\partial x_i} \left( \sum_{k=1}^{L} \frac{\partial}{\partial x_k} \right) \right) f(x) \bigg|_{x = \bar{x}}
\]

Using the expression for \( P_x \) from equation (2-103),

\[
\sum_{i,j=1}^{L} (\sigma_i \sigma_j^T) = 2 \cdot \sum_{i=1}^{L} (\sigma_i \sigma_i^T) = 2P_x
\]

Substituting the result back in equation (2-111)

\[
\frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \frac{1}{2!} D_{\bar{x}}^{2} f = \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \frac{1}{2!} \left( \sum_{i,j=1}^{L} \frac{\partial}{\partial x_i} \left( \sum_{k=1}^{L} \frac{\partial}{\partial x_k} \right) \right) f(x) \bigg|_{x = \bar{x}}
\]

\[
= \frac{1}{2} (\Delta P_x \Delta^T) f(x)|_{x = \bar{x}}
\]

Where, \( \Delta = \frac{\partial}{\partial x} \) is a row vector and \( P_x \) is the covariance matrix, same as defined in equation (2-95).
Using the expression obtained in (2-113), the predicted mean by UT, of equation (2-108), is further simplified to,

$$\bar{y}_{UT} = f(\bar{x}) + \frac{1}{2}(\Delta P_x \Delta^T)f(x)|_{x=\bar{x}} + \frac{1}{2(L+\lambda)} \sum_{i=1}^{2L} \left( \frac{1}{4!} D_{\theta i}^4 f + \frac{1}{6!} D_{\theta i}^6 f + \ldots \right)$$ \hspace{1cm} (2-114)

On comparing equation (2-97) and (2-114), one could clearly make out that the true posterior mean $\bar{y}$ and the mean predicted by UT $\bar{y}_{UT}$, matches exactly to third order and the errors are only introduced in fourth and higher orders. Although the uncertainties considered here are symmetrically distributed about their mean, in case of non-symmetric input uncertainties (with non-zero higher order moments) the approximations are still accurate at least to the second order (as in this case the third order moment won't be zero) with partially higher order moment matching can be obtained, based on the choice of the scaling parameters $\lambda$ as well as the higher-order derivatives of $f(x)$.

In contrast, the EKF approach calculates the posterior mean as,

$$\bar{y}_{LIN} = f(\bar{x})$$ \hspace{1cm} (2-115)

Hence, can be seen that this agrees only to the first order of true posterior mean as obtained in equation (2-97).

**Accuracy of the Covariance**

In this section, first the true covariance of the nonlinear system will be evaluated followed by the covariance of the posterior distribution by UT. And then, the comparison between accuracy of covariance evaluation by UT and by linearization approach is presented, along with some remarks.

The covariance of the nonlinear system can be written as:

$$P_y = E[(y - \bar{y})(y - \bar{y})^T]$$ \hspace{1cm} (2-116)

Using the equations (2-82) and (2-93),

\[
y - \bar{y} = \left[ f(\bar{x}) + D_x f + \frac{1}{2!} D_{xx}^2 f + \frac{1}{3!} D_{xxx}^3 f + \ldots \right] \\
- \left[ f(\bar{x}) + \frac{1}{2!} E[D_{xx}^2 f] + \frac{1}{4!} E[D_{xxx}^3 f] + \ldots \right] \\
= \left[ D_x f + \frac{1}{2!} D_{xx}^2 f + \frac{1}{3!} D_{xxx}^3 f + \ldots \right] - \left[ \frac{1}{2!} E[D_{xx}^2 f] + \frac{1}{4!} E[D_{xxx}^3 f] + \ldots \right]
\] \hspace{1cm} (2-117)

After substituting this expression in equation (2-116) and using the same reasoning as used before, it can be seen that the odd powered terms in the expected value, evaluate to zero.
\[ P_y = E \left[ \left( \frac{D_x f}{2!} + \frac{D_x^2 f}{3!} + \frac{D_x^3 f}{4!} + \ldots \right) - \left( \frac{1}{2!} E [D_x^2 f] + \frac{1}{4!} E [D_x^4 f] + \ldots \right) \right] \{\ldots\}^T \]

\[ P_y = E[(D_x f)(D_x f)^T] + E \left[ \left( \frac{D_x f}{2!} \right)^T \right] + E[\ldots]^T \]

\[ + E \left[ (D_x f)(\frac{D_x^2 f}{2!})^T \right] + \frac{(D_x^2 f)(D_x f)^T}{3!} + \frac{(D_x f)(D_x^2 f)^T}{2!} \]

\[ - E \left[ (D_x f)^2 \right] - E[\ldots]^T - \left( \frac{1}{4!} \right) \frac{(D_x f)^T}{2!} \]

\[ + E \left[ (D_x^2 f)^T \right] + \left( \frac{(D_x f)^T}{2!} \right)^2 \]

\[ = 0 \quad (2-118) \]

Again considering the odd term and using the result of equation (2-90),

\[ E \left[ (D_x f)(D_x f)^T \right] = \frac{1}{2!} E[(D_x f)(D_x f)^T] \]

\[ = 0 \quad (2-119) \]

The first term in equation (2-118) can be expanded as,

\[ E[(D_x f)(D_x f)^T] = E \left[ \left( \sum_{i=1}^{L} \bar{x}_i \frac{\partial}{\partial x_i} f(x) \right) \right] \{\ldots\} \}

\[ = E \left[ \sum_{i,j=1}^{L} \frac{\partial f(x)}{\partial x_i} \bar{x}_i \bar{x}_j \left( \frac{\partial f(x)}{\partial x_j} \right)^T \right] \]

\[ = \sum_{i,j=1}^{L} A_i E[\bar{x}_i \bar{x}_j] A_j \]

\[ = \sum_{i,j=1}^{L} A_i P_{ij} A_j \]

Where \( A_i \) represents the \( i^{th} \) column of the Jacobian matrix \( A \), which is defined as

\[ A = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} |_{x = \bar{x}} & \frac{\partial f(x)}{\partial x_2} |_{x = \bar{x}} & \frac{\partial f(x)}{\partial x_3} |_{x = \bar{x}} & \ldots \end{bmatrix} \]

\[ (2-121) \]

And \( P_{ij} \) is the same element as defined in previous section. Again using the vector definition of equation (2-95).

\[ E[(D_x f)(D_x f)^T] = AP_x A^T \]

\[ (2-122) \]

Therefore, by substituting the results of the equation (2-119) and (2-122) back in equation (2-118), the expression for the true covariance matrix of the nonlinear system can be obtained.
\[ P_y = A P_x A^T + E \left[ \frac{(D_x f)(D_x^2 f)^T}{3!} + \frac{(D_x^2 f)(D_x^2 f)^T}{2!} + \frac{(D_{\bar{x}} f)(D_x f)^T}{3!} \right] + E[...]^T + E \left[ \frac{(D_x f)(D_x^2 f)^T}{2!} \right] + E \left[ \frac{(D_{\bar{x}} f)(D_x f)^T}{4!} \right] + E[...] + \ldots \] (2-123)

In a similar way as that of the mean, covariance of UT is obtained from equation (2-101).

\[ \begin{aligned} (P_y)_{UT} &= \sum_{i=0}^{2L} W_i^{(c)} (Y_i - \bar{y}_{UT})(Y_i - \bar{y}_{UT})^T \\ &= W_0^{(c)} (f(x_0) - f(\tilde{x}))(f(x_0) - f(\tilde{x}))^T \\ &+ \sum_{i=1}^{2L} W_i^{(c)} (f(\tilde{x}_i) - \bar{y}_{UT})(f(\tilde{x}_i) - \bar{y}_{UT})^T \\ &= W_0^{(c)} (f(\tilde{x}) - f(\tilde{x}))(f(\tilde{x}) - f(\tilde{x}))^T \\ &+ \sum_{i=1}^{2L} W_i^{(c)} (f(\tilde{x}_i) - \bar{y}_{UT})(f(\tilde{x}_i) - \bar{y}_{UT})^T \\ &= \frac{1}{2(L+\lambda)} \sum_{i=1}^{2L} \left[ D_{\sigma i} f + \frac{1}{2!} D_{\sigma i}^2 f + \frac{1}{3!} D_{\sigma i}^3 f + \frac{1}{4!} D_{\sigma i}^4 f + \ldots \\ &- \frac{1}{2(L+\lambda)} \sum_{j=1}^{2L} \left( \frac{1}{2!} D_{\sigma j}^2 f + \frac{1}{4!} D_{\sigma j}^4 f + \frac{1}{6!} D_{\sigma j}^6 f + \ldots \right) \right] \right] + \ldots \right] \] (2-124)

On multiplying the expressions,

\[ \begin{aligned} (P_y)_{UT} &= \frac{1}{2(L+\lambda)} \sum_{i=1}^{2L} \left\{ (D_{\sigma i} f)(\ldots)^T + \left[ (D_{\sigma i} f) \left( \frac{1}{2!} D_{\sigma i}^2 f \right)^T \right] + \ldots \right\} \\ &+ \frac{1}{2!} \left( D_{\sigma i}^2 f \right)(\ldots)^T + \ldots \right] - \left[ D_{\sigma i} f \left( \frac{1}{2(L+\lambda)} \sum_{j=1}^{2L} \frac{1}{2!} D_{\sigma j}^2 f \right)^T \right] \\ &- \ldots \right] + \frac{1}{4(L+\lambda)} \left[ \sum_{j=1}^{2L} \frac{1}{2!} D_{\sigma j}^2 f \left( \frac{1}{2!} D_{\sigma j}^2 f \right)^T \right] \\ &- \left[ \frac{1}{2!} D_{\sigma i}^2 f \left( \frac{1}{2(L+\lambda)} \sum_{j=1}^{2L} \frac{1}{2!} D_{\sigma j}^2 f \right)^T \right] - \ldots \right] \] (2-125)

Comparison of EKF and UKF for Flight Path Reconstruction in System Identification
Mohit Mehndiratta

Page 26 / 88
All of the odd power terms reduce to zero as $\vec{\sigma}_i = -\vec{\sigma}_{i+L}$ for $i = 1, \ldots, L$, that is
\[
\sum_{i=1}^{2L} \left[ (D_{\vec{\sigma}_i} f) \left( \frac{1}{2!} D_{\vec{\sigma}_i}^2 f \right)^T \right] = \sum_{i=1}^{L} \left[ (D_{\vec{\sigma}_i} f) \left( \frac{1}{2!} D_{\vec{\sigma}_i}^2 f \right)^T \right] + \sum_{i=L+1}^{2L} \left[ (D_{\vec{\sigma}_i} f) \left( \frac{1}{2!} D_{\vec{\sigma}_i}^2 f \right)^T \right] \\
= \sum_{i=1}^{L} \left[ (D_{\vec{\sigma}_i} f) \left( \frac{1}{2!} D_{\vec{\sigma}_i}^2 f \right)^T \right] + \sum_{i=1}^{L} \left[ (D_{\vec{\sigma}_{i+L}} f) \left( \frac{1}{2!} D_{\vec{\sigma}_{i+L}}^2 f \right)^T \right] \\
= \sum_{i=1}^{L} \left[ (D_{\vec{\sigma}_i} f) \left( \frac{1}{2!} D_{\vec{\sigma}_i}^2 f \right)^T \right] - \sum_{i=1}^{L} \left[ (D_{\vec{\sigma}_i} f) \left( \frac{1}{2!} D_{\vec{\sigma}_i}^2 f \right)^T \right] \\
= 0
\] (2-126)

Similar results can be obtained for other odd power terms as well. Therefore,
\[
(P_y)_{UT} = \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \left[ (D_{\vec{\sigma}_i} f)(...)^T + \frac{1}{2!} \left( \frac{1}{2!} D_{\vec{\sigma}_i}^2 f \right)(...)^T \right] \\
+ \frac{1}{4(L + \lambda)^2} \sum_{i=1}^{2L} \left[ \frac{1}{2!} D_{\vec{\sigma}_i}^2 f \left( \frac{1}{2!} D_{\vec{\sigma}_i}^2 f \right)^T \right] \\
- \frac{1}{2!} D_{\vec{\sigma}_i} f \left( \frac{1}{2!} D_{\vec{\sigma}_i}^2 f \right)^T - [...]^T \\
+ \left[ (D_{\vec{\sigma}_i} f) \left( \frac{1}{3!} D_{\vec{\sigma}_i}^3 f \right)^T \right] + [...]^T + ...
\] (2-127)

This is equivalent to writing
\[
(P_y)_{UT} = \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \{(D_{\vec{\sigma}_i} f)(...)^T\} + HOT
\] (2-128)

Where, HOT (Higher order terms) signifies terms to the fourth order and higher. If just the terms up to second order are considered by neglecting HOT and later expanding the equation for covariance in the vector form, as done in equation (2-110), one can get,
\[
(P_y)_{UT} = \frac{1}{2(L + \lambda)} \sum_{i=1}^{2L} \left\{ \left( \frac{\partial f(x)}{\partial x} \right)_{x = \vec{x}_i} \vec{\sigma}_i \right\} \left( \frac{\partial f(x)}{\partial x} \right)_{x = \vec{x}_i} \vec{\sigma}_i^T
\] (2-129)
\[
(P_y)_{UT} = \frac{1}{2(L + \lambda)} \sum_{i,j=1}^{2L} \left( \frac{\partial f(x)}{\partial x} \right)_{x = \vec{x}_i} \sqrt{(L + \lambda)\vec{\sigma}_i^T} \left( L + \lambda \right) \left( \frac{\partial f(x)}{\partial x} \right)_{x = \vec{x}_j}^T
\] (2-130)
Again on comparing equation (2-123) and (2-132), one could clearly see that the true posterior covariance \( \mathbf{P}_y \) and the covariance predicted by UT \( \mathbf{P}_y^{UT} \), matches exactly to third order (that is only terms to fourth and higher order are incorrect).

In Contrast, the EKF approach calculates the posterior covariance by truncating the Taylor series just after first term, as

\[
\mathbf{P}_y^{LIN} = A\mathbf{P}_xA^T + \text{HOT}
\]  

(2-133)

Hence, one can notice here that linearized approximation has the same approximation order as that of UT, that means it is correct up to third order as well. However, the magnitude of the error in \( \mathbf{P}_y^{UT} \) would be smaller than the error in \( \mathbf{P}_y^{LIN} \), because UT at least contains correctly signed terms to the fourth power and higher, whereas the linear approximation does not contain any term other than the first order term \( A\mathbf{P}_xA^T \).

2.4.1 Unscented Kalman Filter

The Unscented Kalman Filter (UKF) is an extension of the Unscented Transform (UT) to the recursive estimation of the states. The basic difference between EKF and UKF lies in the propagation of Gaussian Random Variable (GRV), through the system dynamics. As known from (Simon J. Julier, 1995) “In EKF the state distribution is approximated by a GRV which is then propagated analytically through the first order linearization of the nonlinear system”. Furthermore, on account of the consideration of the linear terms only in case of EKF, large truncation errors may arise which could later lead to suboptimal performance or sometimes divergence of the filter.

On the other hand UKF is based on the intuition that “With a fixed number of parameters it should be easier to approximate a Gaussian Distribution than it is to approximate an arbitrary nonlinear function” as mentioned in (Simon J. Julier, 1995). Following this intuition, a minimal set of sample (Sigma) points are carefully chosen that completely captures the true mean and covariance of the state distribution (that is, first and second order moments of the prior distribution), and when propagated through the true nonlinear system, they capture the posterior mean and covariance exactly to third order for any nonlinearity. This is better than EKF which only estimates the mean accurately to first order and covariance to third order.

Additional advantage of using UKF is that, no explicit calculations of Jacobians or Hessians are necessary to implement the algorithm. But moreover, the overall number of computations is of the same order as that of EKF, as in the general case of UKF (described in the next section), extra sigma points are to be computed for each propagation of the state vector, due
to its augmentation with noise variables and therefore leads to same amount of computational burden.

### 2.4.1.1 UKF Algorithm for a General Case

A special notation is used here in representing the sigma points, that is, in $\xi^k_{k|k-1}$ the superscript means that it is the sigma point about $x$ whereas, subscript $k|k-1$ means that the sigma point is evaluated at time $k$, when the measurements are available till time $k-1$. In the same way, $\xi^w_{k|k-5}$ would mean that the sigma point about $w$ (process noise) are evaluated at time $k$ using all the measurements available till time $k-5$.

1. The $L$-state continuous time dynamic system is given as,

$$
\begin{align*}
\dot{x} &= f(x(t), u(t), w(t)) \\
z &= h(x(t), u(t), \nu(t)) \\
w(t) &\sim (0, Q(t)) \\
\nu(t) &\sim (0, R(t))
\end{align*}
$$

2. The UKF is initialized as follows.

$$
\begin{align*}
\hat{x}_{0}^+ &= E[x_0] \\
P_{0}^+ &= E[(x_0 - \hat{x}_{0}^+)(x_0 - \hat{x}_{0}^+)^T] \\
\hat{x}_{0}^{a+} &= [\hat{x}_{0}^{+T} 0 0]^T \\
P_{0}^{a+} &= E[(x_0^{a+} - \hat{x}_{0}^{a+})(x_0^{a+} - \hat{x}_{0}^{a+})^T] \nonumber \\
&= \begin{bmatrix} P_0^+ & 0 & 0 \\
0 & Q_0 & 0 \\
0 & 0 & R_0 \end{bmatrix}\nonumber 
\end{align*}
$$

Where, $\hat{x}_{k}^a$ is the augmented form of the state vector which is the vertical concatenation of the original state vector, process noise vector and measurement noise vector; and is used as,

$$
\hat{x}_{k}^a = [\hat{x}_{k}^T \ w_{k}^T \ \nu_{k}^T]^T
$$

Also, here $L$ represents the dimension of the augmented state vector.

3. The following time update equations are used to propagate the state estimate and covariance from one measurement time to the next, till the end of the sampling time $N$.

For $k \in \{1, ..., N-1\}$

Sigma points are calculated as,

$$
\begin{align*}
\xi^a_{k-1} &= \begin{bmatrix} \hat{x}_k^a & \hat{x}_k^a & 0 \\
0 & 0 & 0 \end{bmatrix} + \gamma \sqrt{P^a_{k-1}} \begin{bmatrix} \hat{x}_k^a & \hat{x}_k^a & 0 \\
0 & 0 & 0 \end{bmatrix} - \gamma \sqrt{P^a_{k-1}}
\end{align*}
$$
Where, \( \mathbf{x}_{k-1}^a \) is the augmented sigma point vector and is of the form,

\[
\mathbf{x}_{k}^a = [(\mathbf{x}_{k}^c)^T (\mathbf{x}_{k}^w)^T (\mathbf{x}_{k}^n)^T]^T
\]

Also, the scaling parameters \( \gamma \) and \( \lambda \) are

\[
\gamma = \sqrt{L + \lambda} \\
\lambda = \alpha^2(L + \kappa) - L
\]

and \( \alpha, \kappa \) and \( \beta \) are same as defined in UT

The time update equations are:

\[
\mathbf{x}_{k|k-1} = f(\mathbf{x}_{k-1}^c, \mathbf{u}_{k-1}, \mathbf{x}_{k-1}^w)
\]

\[
\hat{\mathbf{x}}_k = \sum_{i=0}^{2L} W_i^{(m)} \mathbf{x}_{i,k|k-1}
\]

\[
P_k^c = \sum_{i=0}^{2L} W_i^{(c)} (\mathbf{x}_{i,k|k-1} - \hat{\mathbf{x}}_k) (\mathbf{x}_{i,k|k-1} - \hat{\mathbf{x}}_k)^T
\]

\[
\mathbf{y}_{k|k-1} = h(\mathbf{x}_{k|k-1}^c, \mathbf{u}_k, \mathbf{x}_{k|k-1}^w)
\]

\[
\bar{\mathbf{y}}_k = \sum_{i=0}^{2L} W_i^{(m)} \mathbf{y}_{i,k|k-1}
\]

The measurements update equations are:

\[
P_{\bar{\mathbf{y}}_k \bar{\mathbf{y}}_k} = \sum_{i=0}^{2L} W_i^{(c)} (\mathbf{y}_{i,k|k-1} - \bar{\mathbf{y}}_k) (\mathbf{y}_{i,k|k-1} - \bar{\mathbf{y}}_k)^T
\]

\[
P_{\mathbf{x}_k \mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} (\mathbf{x}_{i,k|k-1} - \hat{\mathbf{x}}_k) (\mathbf{y}_{i,k|k-1} - \bar{\mathbf{y}}_k)^T
\]

The update equations are similar to the standard Kalman filter equations. The only change here is in the computation of the Kalman gain matrix \( K_k \), which is calculated using the output covariance matrix \( P_{\bar{\mathbf{y}}_k \bar{\mathbf{y}}_k} \) and cross covariance matrix \( P_{\mathbf{x}_k \mathbf{y}_k} \) instead of the output matrix \( C_k \), process noise covariance matrix \( Q_k \) and measurement noise covariance matrix \( R_k \), as already seen in equation (2-57). Moreover, an equivalence within the two expressions is easy obtain and hence presented in the appendix A.5.

\[
K_k = P_{\mathbf{x}_k \mathbf{y}_k} P_{\bar{\mathbf{y}}_k \bar{\mathbf{y}}_k}^{-1}
\]

\[
\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k + K_k (\mathbf{z}_k - \bar{\mathbf{y}}_k)
\]

\[
P_k^c = P_k^c - K_k P_{\bar{\mathbf{y}}_k \bar{\mathbf{y}}_k} K_k^T
\]
Where the weights $W_i$ are given by,

$$W_0^{(m)} = \frac{\lambda}{L + \lambda}$$

$$W_0^{(c)} = \frac{\lambda}{L + \lambda} + 1 - \alpha^2 + \beta$$

$$W_i^{(m)} = W_i^{(c)} = \frac{1}{2(L + \lambda)} \quad i = 1, \ldots, 2L$$

### 2.4.1.2 Variation in Implementation

For a special case which is often encountered in practice where both process and measurement noise are purely additive, the computational complexity of the UKF algorithm can be reduced. In such a situation, the state vector need not be augmented by noise RV’s. This in turn reduces the dimension of sigma points as well as the number of sigma points used. The covariance of the noise sources ($Q$ and $R$) are then incorporated into the state covariance using a simple additive procedure. The complexity of the algorithm is of order $L^3$ (where $L$ is the dimension of original state vector). This is the same complexity as that of EKF. The detailed procedure can be seen in (Hayking, 2001).

The Algorithm is given as,

1. The $L$-state continuous time dynamic system is given as,

$$\dot{x} = f(x(t), u(t), t) + w(t)$$

$$z = h(x(t), u(t), t) + \nu(t)$$

$$w(t) \sim (0, Q(t))$$

$$\nu(t) \sim (0, R(t))$$

(2-151)

2. Initialization:

$$\tilde{x}_0^+ = E[x_0]$$

$$P_0^+ = E[(x_0 - \tilde{x}_0^+)(x_0 - \tilde{x}_0^+)^T]$$

(2-152)

3. Definition of Sigma points:

$$X_0 = \tilde{x}$$

$$X_i = \tilde{x} + (\gamma \sqrt{P_x})_i \quad i = 1, \ldots, L$$

$$X_i = \tilde{x} - (\gamma \sqrt{P_x})_{i-L} \quad i = L + 1, \ldots, 2L$$

(2-153)

Here, $L$ represents the dimension of the original state vector and

$$\gamma = \sqrt{L + \lambda}$$

$$\lambda = \alpha^2(L + \kappa) - L$$

(2-154)
4. The time update equations, till the end of the sampling time $N$.

For $k \in \{1, \ldots, N - 1\}$

$$\mathbf{x}_{k|k-1} = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$$  \hfill (2-155)

$$\hat{\mathbf{x}}_k = \sum_{i=0}^{2L} W_i^{(m)} \chi_{i,k|k-1}$$  \hfill (2-156)

$$P_k^{-} = \sum_{i=0}^{2L} W_i^{(c)} (\chi_{i,k|k-1} - \hat{\chi}_k^{-}) (\chi_{i,k|k-1} - \hat{\chi}_k^{-})^T + Q_k$$  \hfill (2-157)

$$Y_{k|k-1} = h(\mathbf{x}_{k|k-1}, \mathbf{u}_k)$$  \hfill (2-158)

$$\hat{\mathbf{y}}_k^{-} = \sum_{i=0}^{2L} W_i^{(m)} Y_{i,k|k-1}$$  \hfill (2-159)

Measurement update

$$P_{\mathbf{y}_k\hat{\mathbf{y}}_k} = \sum_{i=0}^{2L} W_i^{(c)} (Y_{i,k|k-1} - \hat{\mathbf{y}}_k^{-}) (Y_{i,k|k-1} - \hat{\mathbf{y}}_k^{-})^T + R_k$$  \hfill (2-160)

$$P_{\mathbf{x}_k\mathbf{y}_k} = \sum_{i=0}^{2L} W_i^{(c)} (\chi_{i,k|k-1} - \hat{\chi}_k^{-}) (Y_{i,k|k-1} - \hat{\mathbf{y}}_k^{-})^T$$  \hfill (2-161)

$$K_k = P_{\mathbf{x}_k\mathbf{y}_k} P_{\mathbf{y}_k\hat{\mathbf{y}}_k}^{-1}$$  \hfill (2-162)

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k (\mathbf{z}_k - \hat{\mathbf{y}}_k^-)$$  \hfill (2-163)

$$P_k^+ = P_k^- - K_k P_{\mathbf{y}_k\hat{\mathbf{y}}_k} K_k^T$$  \hfill (2-164)

Where weights $W_i$'s are same as defined in section 2.4.1.1.

Beside this formulation of the weights, there is another formulation as mentioned in (Simon J. Julier, 1995). It is based upon $\kappa$ as the only scaling parameter of the sigma points about the mean $\bar{x}$. The expression looks like,

$$W_0^{(m)} = \frac{\kappa}{L + \kappa}$$

$$W_0^{(c)} = \frac{\kappa}{L + \kappa}$$  \hfill (2-165)

$$W_i^{(m)} = W_i^{(c)} = \frac{1}{2(L + \kappa)}$$ \quad $i = 1, \ldots, 2L$

And rest of the algorithm is same but $\kappa$ is used everywhere instead of $\lambda$ in equations (2-151) to (2-164).
2.4.2 Unscented Rauch, Tung and Striebel Smoother

In a fixed-interval smoothing problem, the entire set of observations over the interval are available to be used to estimate the entire set of states within that interval. A two filter formulation, one running forward in time and other running backward, may be employed here as well. The smoothed estimate may then be expressed as a linear combination of the forward and the backward estimate. For one case, the UKF can be employed as a forward filter whereas Backward Sigma-Point Information Filter can be selected to be the backward one as mentioned in (Gabriel Terejanu).

The application of UKF as a forward filter is similar to as mentioned in section 2.4.1. Furthermore, during the forward pass the values of mean and covariance will be stored, to be used later during the smoothing step. One other expression that is also calculated and stored during the forward pass

\[ P_{\hat{x}_k}^{f+} = \sum_{i=0}^{2n} W_i^{(c)} (\chi_{i,k|k}^f - \hat{x}_k^f) (\chi_{i,k|k+1}^f - \hat{x}_{k+1}^f)^T \]  

(2-166)

Here in \( \chi_{i,k|k+1}^f \) *’ signifies that the sigma point doesn’t include the contribution of the process noise \( w_k \). Also, \( f, b, s \) signify the forward pass, backward pass and smoothed estimate as used previously in case of Extended RTS smoother. Firstly, the Sigma-Point Information Filter (SPIF) is presented; so as to be able to derive later, the Backward Sigma-Point Information Filter (BSPIF).

Sigma-Point Information Filter (SPIF)

To be able to begin the filter recursions for the backward pass of a smoother, a useful substitution is done, since not much information is available at that time. The idea behind the substitution is that "The Kalman filter recursively propagates the covariance of the state estimates, while the information filter propagates its inverse", as mentioned in (Gabriel Terejanu). Using this idea, the weighted statistical linearization (details can be seen in (Gabriel Terejanu)) of the system can be used to obtain,

\[ x_{k+1} = A_k x_k + f_k + w_k + e_k^{(1)} \]  

(2-167)

Where,

\[ A_k = P_{\hat{x}_k}^{f+} \chi_{k+1}^{f-} (P_k^f)^{-1} \]  

(2-168)

\[ f_k = \hat{x}_k^{f-} - A_k \hat{x}_k^f \]  

(2-169)

\[ P_{e_k e_k}^{(1)} = P_{k+1}^{f-} - A_k P_k^f A_k^T \]  

(2-170)

And \( e_k^{(1)} \) is the state linearization error. Also, \( P_{k+1}^{f-} \) is known as the forecast error covariance and it only captures the error in the state transition linearization but does not account for the effect of process noise.

The measurement equation is also linearized based on the same principle.

\[ z_k = C_k \hat{x}_k + h_k + v_k + e_k^{(2)-} \]  

(2-171)
Where,

\[
C_k^r = P_{x_k|y_k}^f (P_k^f)^{-1}
\]  \hspace{1cm} (2-172)

\[
h_k^r = \hat{y}_k^f - C_k^r \hat{x}_k^f
\]  \hspace{1cm} (2-173)

\[
P_{e_k|e_k}^{(2)-} = P_{y_k|y_k}^f - C_k^r P_k^f (C_k^r)^T
\]  \hspace{1cm} (2-174)

Where, \(e_k^{(2)-}\) is the measurement linearization error and \(P_{y_k|y_k}^f\) is the output covariance matrix that does not include the effect of measurement noise and is defined based on equation (2-160) as,

\[
P_{y_k|y_k}^f = \sum_{i=0}^{2L} W_i^{(c)} (Y_{i,k|k-1} - \hat{y}_k^-) (Y_{i,k|k-1} - \hat{y}_k^-)^T
\]  \hspace{1cm} (2-175)

As mentioned above, here the Backward Sigma-Point Information Filter (BSPIF) is used as the backward filter and its brief derivation is given below, but for the detailed derivation one can refer to (Gabriel Terejanu).

**Backward Sigma-Point Information Filter (BSPIF)**

To derive unscented backward information filter, weighted statistical linearization is used, details of which again can be seen in (Gabriel Terejanu). Given the following backward Markovian model (in which future states only depends on the present state values), the backward forecast (also called as hindcast) can be estimated as:

\[
x_k = A_k^{b+1} \hat{x}_{k+1} - A_k^{-1} f_k + W_{k+1}^b
\]  \hspace{1cm} (2-176)

Where,

\[
A_k^{b+1} = A_k^{-1} \left[ I - \left( Q_k + P_{e_k e_k}^{(1)} \right) \Pi_{k+1}^{-1} \right]
\]

\[
Q_{k+1}^{b+1} = A_k^{b+1} \left( Q_k + P_{e_k e_k}^{(1)} \right) (A_k^{-1})^T
\]

\[
\Pi_{k+1} = A_k \Pi_{k} A_k^T + Q_k + P_{e_k e_k}^{(1)}
\]

\[
\Pi_0 = P_0
\]

Here it is assumed that \(\Pi_{k+1}\) increases at a very fast rate that is, \(det(\Pi_{k+1}) \rightarrow \infty\), due to large initial uncertainty and on account of this the contribution of \(\Pi_{k+1}^{-1}\) is neglected, resulting in \(A_k^{b+1} \approx A_k^{-1}\) and a simplified backward model.

\[
\hat{x}_k^{b-} = A_k^{-1} (\hat{x}_{k+1}^{b+} - f_k)
\]  \hspace{1cm} (2-178)

It is assumed that the quantities \(A_k, f_k\) and \(P_{e_k e_k}^{(1)}\) are already determined and stored during the forward pass.
Using equation (2-167), the true backward estimate can be obtained as
\[ x_k^b = A_k^{-1} x_{k+1}^b - A_k^{-1} f_k - A_k^{-1} w_k - A_k^{-1} e^{(1)}_k \]  (2-179)

Hence, the hindcast error would be
\[ \hat{e}_k^b = x_k^b - \bar{x}_k^b = A_k^{-1} x_{k+1}^b - A_k^{-1} f_k - A_k^{-1} w_k - A_k^{-1} e^{(1)}_k - A_k^{-1} (\hat{x}_{k+1}^b - f_k) \]  (2-180)

Also, the error covariance matrix
\[ P_k^b = E \left[ (\hat{e}_k^b - \bar{e}_k^b)^T \right] \]  (2-181)

With some matrix manipulations (not presented here), the final expression for the error covariance matrix can be obtained as,
\[ P_k^b = A_k^{-1} \left( P_{k+1}^b + Q_k + P_{e_k e_k}^{(1)} \right) (A_k^{-1})^T \]  (2-182)

Given the forward pass results, sigma points for the linearization of the measurement update can be regenerated for better accuracy.
\[ z_k = C_k^+ x_k + h_k^+ + v_k + e_k^{(2)+} \]  (2-183)

Where,
\[ C_k^+ = (p_{x_k y_k}^f)^T (p_k^f)^{-1} \]  (2-184)
\[ h_k^+ = \hat{y}_k^f - C_k^+ \hat{x}_k^{f+} \]  (2-185)
\[ P_{e_k e_k}^{(2)+} = p_{y_k y_k}^f - C_k^+ p_k^f (C_k^+)^T \]  (2-186)

And \( p_{y_k y_k}^f \) is same as defined in equation (2-175),

The other quantities like \( \hat{y}_k^f, p_{x_k y_k}^f \) are computed in the same way as described in section 2.4.1. As done with the process model, quantities \( C_k^+, h_k^+, P_{e_k e_k}^{(2)+} \) are also determined and stored during the forward pass.

The backward state estimate after assimilating the current measurement is computed as,
\[ \hat{x}_k^{b+} = \hat{x}_k^b + K_k^b (z_k - \bar{y}_k^b) \]  (2-187)

Where \( z_k \) are the measurements obtained at time instant \( k \), and
\[ \bar{y}_k^b = C_k^+ \hat{x}_k^b + h_k \]  (2-188)
Now the error and covariance of the backward estimate are given by,

\[ \hat{e}_{k}^{b+} = \hat{e}_{k}^{b-} - K_{k}^{b} \left( \hat{e}_{k}^{b-} + v_{k} + e_{k}^{(2)+} \right) \]  \hfill (2-189)

\[ P_{k}^{b+} = P_{k}^{b-} + K_{k}^{b} \left( C_{k}^{b} P_{k}^{b-} (C_{k}^{b})^{T} + R_{k} + P_{e_{k} e_{k}}^{(2)+} \right) (K_{k}^{b})^{T} - P_{k}^{b-} (C_{k}^{b})^{T} (K_{k}^{b})^{T} \]  \hfill (2-190)

The backward gain \( K_{k}^{b} \) can be determined in a similar way as done in EKF, that is by minimizing the trace of the backward error covariance \( P_{k}^{b+} \). That is,

\[ \frac{\partial}{\partial K_{k}^{b}} tr(P_{k}^{b+}) = 0 \]  \hfill (2-191)

The resulting expressions looks like,

\[ K_{k}^{b} = P_{k}^{b-} (C_{k}^{b})^{T} (C_{k}^{b} P_{k}^{b-} (C_{k}^{b})^{T} + R_{k} + P_{e_{k} e_{k}}^{(2)+})^{-1} \]  \hfill (2-192)

Substituting it back in (2-190) the backward error covariance becomes,

\[ P_{k}^{b+} = (I - K_{k}^{b} C_{k}^{b}) P_{k}^{b-} \]  \hfill (2-193)

For the information form of this filter, the backward estimates and their respective covariance matrices are replaced by the associated information states and information matrices. That is,

\[ \hat{i}_{k}^{b+} = I_{k}^{b+} \hat{x}_{k}^{b+} \quad \hat{i}_{k}^{b-} = I_{k}^{b-} \hat{x}_{k}^{b-} \]  \hfill (2-194)

are known as the Information states.

And,

\[ I_{k}^{b+} = (P_{k}^{b+})^{-1} \quad I_{k}^{b-} = (P_{k}^{b-})^{-1} \]  \hfill (2-195)

are known as the Information matrices.

By substituting these relations back in equations (2-178), (2-182), (2-187) and (2-193), the following expressions can be obtained after applying matrix inversion lemma and some algebra as given in (Gabriel Terejanu),

\[ I_{k}^{b-} = A_{k}^{T} (I - K_{k}^{b}) I_{k+1}^{b+} A_{k} \]  \hfill (2-196)

\[ I_{k}^{b+} = I_{k}^{b-} + (C_{k}^{b})^{T} \left( R_{k} + P_{e_{k} e_{k}}^{(2)+} \right)^{-1} C_{k}^{b} \]  \hfill (2-197)

\[ I_{k}^{b+} = I_{k}^{b-} + (C_{k}^{b})^{T} \left( R_{k} + P_{e_{k} e_{k}}^{(2)+} \right)^{-1} C_{k}^{b} \]  \hfill (2-197)

Where,

\[ K_{k}^{b} = I_{k+1}^{b+} \left[ P_{k+1}^{b+} + \left( Q_{k} + P_{e_{k} e_{k}}^{(1)} \right)^{-1} \right]^{-1} \]  \hfill (2-198)
The hindcast estimate equation (2-178) becomes the backcast information estimate equation.

\[ \hat{b}_k^b = I_k^b \ A_k^{-1} \left( (I_{k+1}^b)^{-1} \hat{t}_{k+1}^b - f_k \right) \]  

(2-200)

Substituting equation (2-196) in (2-200),

\[ \hat{b}_k^b = A_k^T (I - K_k^b) (I_{k+1}^b)^{-1} \hat{t}_{k+1}^b - f_k \]  

(2-201)

Similarly, the backward information estimate can be obtained from equation (2-187) as,

\[ \hat{b}_k^b = \hat{t}_k^b + I_k^b K_k^b (z_k - h_k^b) \]  

(2-202)

Using equation (2-193)

\[ P_k^b = (I - K_k^b C_k^b) P_k^b \]  

(2-203)

After rearranging the terms,

\[ K_k^b = (I - P_k^b (P_k^b)^{-1}) (C_k^b)^{-1} \]

\[ = (I - (T_k^b)^{-1} I_k^b) (C_k^b)^{-1} \]  

(2-204)

Substituting equation (2-204) in (2-202)

\[ \hat{b}_k^b = \hat{t}_k^b + I_k^b \left( I - (T_k^b)^{-1} I_k^b \right) (C_k^b)^{-1} (z_k - h_k^b) \]

\[ = \hat{t}_k^b + \left( I_k^b + (T_k^b)^{-1} I_k^b \right) (C_k^b)^{-1} (z_k - h_k^b) \]  

(2-205)

Obtaining an expression for \( I_k^b \) from equation (2-197) and putting in above equation

\[ \hat{b}_k^b = \hat{t}_k^b + \left( I_k^b + (C_k^b)^T (R_k + F^{(2)+}_k)^{-1} C_k^b - I_k^b \right) (C_k^b)^{-1} (z_k - h_k^b) \]

\[ = \hat{t}_k^b + \left( (C_k^b)^T (R_k + F^{(2)+}_k)^{-1} \right) (C_k^b)^{-1} (z_k - h_k^b) \]  

(2-206)

After some matrix transformations, an unbiased smoothed estimate, which is a linear combination of forward and backward estimates, can be used to compute the smoothed error covariance as done in (Gabriel Terejanu),

\[ P_k^s = \left[ (P_k^f)^{-1} + (P_k^b)^{-1} \right]^{-1} \]  

(2-207)

Using matrix inversion lemma and further simplification as mentioned in (Gabriel Terejanu), the following expressions can be obtained,

\[ K_k^s = P_k^f I_k^b (I + P_k^f I_k^b)^{-1} \]  

(2-208)

\[ P_k^s = (I - K_k^s) P_k^f \]  

(2-209)
\[ x_k^+ = (I - K_k^b)x_k^f + P_k^b k_k^b \] (2-210)

Intuitively, one can notice that if the estimate obtained during the forward pass is ‘good’, that is, if the value of covariance is ‘smaller’ than the one obtained during backward pass, the resulting weight \( K_k^b \) will be rather small, this means that the estimate \( x_k^f \) will be weighted more heavily in obtaining \( x_k^s \) from equation (2-210), as seen in case of Extended RTS as well.

As already stated for Extended RTS, this type of two filter formulation is generally more demanding in terms of computational power and Rauch, Tung and Striebel introduced a more computationally efficient form for fixed interval smoothing that combines the backward filter and smoothing step into a single-step correction to the forward filter estimate. This formulation is based on the information gain as obtained from the forward pass. To be consistent with the nomenclature of Extended RTS, now onwards the gain matrix \( K_k \) will be replaced with \( M_k \) and the following assumptions are made,

\[
C_k^+ \approx C_k^- = C_k
\]

\[ h_k^+ \approx h_k^- = h_k \] (2-211)

\[ p_{e_k e_k}^{(2)+} \approx p_{e_k e_k}^{(2)-} = p_{e_k e_k}^{(2)} \]

These assumptions are valid as long as the difference between the prior state estimate and the updated (or posterior) state estimate is small. Also, they can be violated in highly nonlinear regions of certain process models for example bifurcation manifolds. But here sufficient smoothness of the process model for linearization is assumed to justify these assumption, as done in (Gabriel Terejanu).

By applying the matrix inversion lemma in (2-207), the smoothed error covariance can be written as,

\[ P_k^s = P_k^f - P_k^f (P_k^b + P_k^f)^{-1} P_k^f \] (2-212)

Using some matrix simplification and substitutions, the inverse of the sum becomes,

\[ (P_k^b + P_k^f)^{-1} = A_k^T (P_{k+1}^s + P_{k+1}^b)^{-1} A_k \] (2-213)

The only backward information reference available is the backward error covariance, which can be expressed in terms of the smoothed and the forecast error covariance (Information matrix) as,

\[ P_{k+1}^{b+} = \left[ (P_{k+1}^s)^{-1} + I_{k+1}^{-1} \right]^{-1} = \left[ (P_{k+1}^s)^{-1} + (P_{k+1}^f)^{-1} \right]^{-1} \] (2-214)

Hence by further substitution and simplification, one can arrive to an expression for smoothed error covariance. Details can be seen in (Gabriel Terejanu)

\[ P_k^s = P_k^f - M_k^s (P_{k+1}^f - P_{k+1}^s)(M_k^s)^T \] (2-215)
Where,

\[ M_k^s = P_k^s f_k^s (P_{k+1}^f)^{-1} \]  

(2-216)

The similar steps can be used to eliminate the backward information and an expression for smoothed estimate can be obtained (without any backward term). That is,

\[ x_k^s = P_k^s i_k^f + P_k^s i_k^b \]  

(2-217)

The quantities \( i_k^f \) and \( i_{k+1}^b \) are obtained in a similar way as their backward counterparts, as obtained in (Gabriel Terejanu).

Substituting the expression for \( i_k^b \) obtained in (2-200), in the above equation

\[ x_k^s = P_k^s i_k^f + P_k^s i_k^b A_k^{-1} \left[ (i_{k+1}^b)^{-1} i_{k+1}^b - f_k \right] \]  

(2-218)

Incrementing the time step in (2-181) and writing \( i_{k+1}^b \) in terms of \( i_{k+1}^b \) using (2-205)

\[ x_{k+1}^s = P_{k+1}^s i_{k+1}^f + P_{k+1}^s i_{k+1}^b A_{k+1}^{-1} \left[ (i_{k+1}^b)^{-1} (P_{k+1}^s)^{-1} x_{k+1}^s - i_{k+1}^b \right] + f_k \]  

(2-219)

Similarly \( i_{k+1}^b \) can be expressed in terms of \( x_{k+1}^s \) as,

\[ i_{k+1}^b = (P_{k+1}^s)^{-1} x_{k+1}^s - i_{k+1}^b \]  

(2-220)

Substituting the above expressions for \( i_{k+1}^b \) in (2-218)

\[ x_k^s = P_k^s i_k^f + P_k^s i_k^b A_k^{-1} \left[ (i_{k+1}^b)^{-1} (P_{k+1}^s)^{-1} x_{k+1}^s - i_{k+1}^b \right] + f_k \]  

(2-221)

On substituting equation (2-196) for \( i_{k+1}^b \) and equation (2-214) for \( i_{k+1}^b \)

\[ x_k^s = P_k^s i_k^f + P_k^s A_k^T \left[ (I - M_k^s)^{-1} \left( (P_{k+1}^s)^{-1} + (P_{k+1}^f)^{-1} \right)^{-1} \left( (P_{k+1}^s)^{-1} x_{k+1}^s \right) \right] \]  

(2-222)

\[ x_k^s = P_k^s i_k^f + P_k^s A_k^T \left[ (I - M_k^s)^{-1} (P_{k+1}^s)^{-1} x_{k+1}^s + f_k \right] \]  

(2-223)

Therefore,

\[ x_k^s = P_k^s i_k^f + \Phi_k \left[ \left( (P_{k+1}^s)^{-1} + (P_{k+1}^f)^{-1} \right)^{-1} \left( (P_{k+1}^s)^{-1} x_{k+1}^s - i_{k+1}^f \right) \right] \]  

(2-224)
Where,

\[
\Phi_k = M_k^s (P_{k+1}^{f-} - P_{k+1}^s) (P_{k+1}^{f-})^{-1}
\]  \hspace{1cm} (2-225)

After some more substitution and matrix manipulations, finally an expression for smoothed estimate, independent of any term related to backward pass, can be obtained.

\[
x_k^s = \hat{x}_k^{f+} + M_k^s (x_{k+1}^s - \hat{x}_{k+1}^{f-})
\]  \hspace{1cm} (2-226)

The Unscented RTS steps can be summarized as,

A. Initialize the UKF with

\[
\begin{align*}
\hat{x}_0^{f+} &= E[x_0] \\
P_0^{f+} &= E \left[ (x_0 - \hat{x}_0^{f+})(x_0 - \hat{x}_0^{f+})^T \right]
\end{align*}
\]  \hspace{1cm} (2-227)

B. Compute the forward pass and save the quantities: \( \hat{x}_k^{f+}, \hat{x}_k^{f-}, P_k^{f+}, P_k^{f-}, P_{\hat{x}_k^{f+},\hat{x}_k^{f-}} \).

C. Initialize URTS smoother as,

\[
\begin{align*}
x_N^s &= \hat{x}_N^{f+} \\
P_N^s &= P_N^{f+}
\end{align*}
\]  \hspace{1cm} (2-228)

D. Run the combined backward pass and smoothing correction for \( k = N - 1, ..., 1 \)

\[
\begin{align*}
M_k^s &= P_{\hat{x}_k^{f+},\hat{x}_{k+1}^{f-}} (P_{k+1}^{f-})^{-1} \\
P_k^s &= P_k^{f+} - M_k^s (P_{k+1}^{f-} - P_{k+1}^s) (M_k^s)^T \\
x_k^s &= \hat{x}_k^{f+} + M_k^s (x_{k+1}^s - \hat{x}_{k+1}^{f-})
\end{align*}
\]  \hspace{1cm} (2-229, 2-230, 2-231)
2.5 The Model

2.5.1 Dynamics of the System

The dynamic system in consideration is based on the nonlinear equations of motion of the rigid body aircraft, formulated with respect to a body fixed point, whose details are given in (Holzapfel, 2012). The contribution of wind is taken into account and are kept as last three states of the state vector. This leads to the difference in aerodynamic and kinematic velocities i.e., they are no longer same. Therefore, the state vector includes positions, kinematic velocities, Euler attitude angles and the wind velocities. The reference point that is used in the formulation of rigid body equations, is the sensor reference point INS, with the measurements obtained in the sensor INS-frame. The system equations describing the motion of an aircraft are given as,

\[
x = \begin{bmatrix}
(r^G_{WGS84})_B \\
(v^G_K)_B \\
\Psi_{euler} \\
(w^G_W)_B
\end{bmatrix} = \\
\begin{bmatrix}
\frac{(v^G_K)_B}{(h^G_{WGS84})} \\
\frac{u^G_K}{(h^G_{WGS84})} \\
\cos(\Phi^G_{WGS84}) (R_E + (h^G_{WGS84})) \\
-\frac{w^G_W}{(h^G_{WGS84})}
\end{bmatrix} + w_r
\]

\[
\begin{bmatrix}
(a^I_{K})_{B}^I - (w^I_{K})_{B} \\
(v^I_{K})_{B} \\
(\omega^I_{K})_{B} \\
\omega^I_{W}
\end{bmatrix} = f(x, u, w)
\]

Where the respective state vectors are defined as,

\[
(r^G_{WGS84}) = \begin{bmatrix}
(\lambda^G_{WGS84}) \\
(\varphi^G_{WGS84}) \\
(h^G_{WGS84})
\end{bmatrix}; (v^G_K)_B = \begin{bmatrix}
(u^G_K)_B \\
(v^G_K)_B \\
(w^G_K)_B
\end{bmatrix}; \Psi_{euler} = [\Phi, \Theta, \Psi]; (v^G_W)_B = \begin{bmatrix}
(u^G_W)_B \\
(v^G_W)_B \\
(w^G_W)_B
\end{bmatrix}
\]

And \(w_r\), \(w_{\Psi_{euler}}\) and \(w_{\psi_{w}}\) are the random process noise vectors in position differential equations, attitude angle differential equations and wind velocity differential equations respectively. Also \(R_E\) represents the radius of earth.

The position differential equations, which is given in terms of \((v^G_K)_O\), are the integrations of velocities in the respective directions and are usually formulated on basis of WGS84
coordinates. The kinematic velocity \((v^G_K)^E\), is in the North-East-Down (NED) frame and is computed using transformation matrix \(M_{BO}\) as,

\[
(v^G_K)^E = M_{BO}^T \cdot (v^G_K)^B
\]

Where, \(M_{BO}\) is evaluated using equation (2-248).

The commanded inputs to the above differential equations are linear accelerations, rotational rates and rotational accelerations. It is worth noting that, even though the inputs given to the system (aircraft) are control surface deflections, but the input received by the model are these commanded inputs (accelerations, rates and rotational accelerations) and they are of the form,

\[
u = \begin{bmatrix} (a^IINS)_B^H \\ (\omega^I_B)_B \\ (\dot{\omega}^I_B)_B \end{bmatrix}
\]

Also, the measured input is represented as,

\[u_{meas} = u + \tilde{u}
\]

The error model for inputs are mentioned below, that includes the bias terms for gyros \((\Delta \omega^IINS)_B\), accelerometer measurements \((\Delta a^IINS)_B^H\) and white, Gaussian noise (stochastic part of the input). But for the case considered in this report, all the bias terms in the inputs were kept zero and the only source of uncertainty present in the inputs, are the input noise \((w_a, w_\omega, w_\dot{\omega})\).

Furthermore, a misalignment matrix \(M_{INSB}\) between the inertial sensor and body fixed frame is also introduced, as the IMU is considered not to be perfectly aligned with body fixed coordinate frame. The relations between true and measured input quantities are,

\[
\begin{align*}
(a^IINS)_B^{H,meas} &= M_{INS,B}(a^IINS)_B^H + (\Delta a^IINS)_B^H + w_a \\
(\omega^I_B)_{INS,meas} &= M_{INS,B}(\omega^I_B)_B + (\Delta \omega^I_B)_INS + w_\omega \\
(\dot{\omega}^I_B)_{INS,meas} &= M_{INS,B}(\dot{\omega}^I_B)_B + w_\dot{\omega}
\end{align*}
\]

Where,

\[
\tilde{u} = \begin{bmatrix} w_a \\ w_\omega \\ w_\dot{\omega} \end{bmatrix}
\]

The misalignment matrix \(M_{INSB}\) is parameterized in euler angles, as

\[
M_{INSB}(\Delta \Psi_{INS}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \Delta \phi_{INS} & \sin \Delta \phi_{INS} \\ 0 & -\sin \Delta \phi_{INS} & \cos \Delta \phi_{INS} \end{bmatrix} \\
\begin{bmatrix} \cos \Delta \theta_{INS} & 0 & -\sin \Delta \theta_{INS} \\ 0 & 1 & 0 \\ \sin \Delta \theta_{INS} & 0 & \cos \Delta \theta_{INS} \end{bmatrix} \cdot \begin{bmatrix} \cos \Delta \psi_{INS} & \sin \Delta \psi_{INS} & 0 \\ -\sin \Delta \psi_{INS} & \cos \Delta \psi_{INS} & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

(2-241)
The total process noise is then described as

\[ \mathbf{w} = \begin{bmatrix} \mathbf{\tilde{u}} \\ \mathbf{w}_{\text{proc}} \end{bmatrix} \]  

\[ \mathbf{w}_{\text{proc}} : \text{process noise in state vector} \]

Other than GPS and IMU measurements, another measurement source is the Pitot tube, which gives the information about barometric (or pressure) altitude, indicated airspeed \( V_{\text{IAS}} \) and flow angles \( \alpha_A^{NB} \) and \( \beta_A^{NB} \) (index \( A \) represents the location and \( NB \) is the abbreviation for Nose boom) by measuring static and dynamic pressure. The indicated airspeed was computed as the norm of resulting velocity at the Pitot tube position. It is then corrected by an estimate of the air density \( \rho \) (at the flying altitude) and becomes the true airspeed \( V_{\text{TAS}} \). This is done as indicated airspeed is usually computed using the standard sea-level density \( \rho_0 = 1.225 \frac{kg}{m^3} \).

That is,

\[ V_{\text{TAS}} = \sqrt{\frac{\rho_0}{\rho}} \cdot V_{\text{IAS}} \]  

(2-243)

And the error model used for the true airspeed \( V_{\text{TAS}} \), consisting of a bias term \( \Delta V_{\text{TAS}} \), a scale factor error term \( K_V \) and a white measurement noise \( \nu_{\text{TAS}} \), can be given as

\[ V_{\text{TAS, meas}}^{NB} = \frac{(V_{\text{TAS}}^{NB} - \Delta V_{\text{TAS}}) \cdot (K_V + 1) + \nu_{\text{TAS}}}{V_{\text{TAS}}} \]  

(2-244)

Here also, except of the \( \nu_{\text{TAS}} \), both the bias and the scale factor were kept zero.

All the measured rates and accelerations are obtained in the Body fixed frame and are later transformed to North-East-Down (NED) frame using the transformation matrix \( \mathbf{M}_{BO} \). Not all, but some of the output transformations are mentioned below. For details one can refer (Holzapfel, 2012).

\[ \left( \begin{array}{c} v_{K}^{G}_{\text{GPS}} \end{array} \right)^E_{0,\text{GPS}} = \mathbf{M}_{BO}^T \left( \left( \begin{array}{c} v_{K}^{G} \end{array} \right)^E_B + \left( \begin{array}{c} \mathbf{\omega}^{IB} \end{array} \right)_B \times \left( \mathbf{r}^{G,\text{GPS}} \right)_B \right) \]  

(2-245)

\[ \begin{bmatrix} V_{\text{TAS}}^{NB} \\ \alpha_A^{NB} \\ \beta_A^{NB} \end{bmatrix} = \begin{bmatrix} \sqrt{(u_A^{NB})_B^E + (v_A^{NB})_B^E + (w_A^{NB})_B^E} \\ \arctan \frac{(w_A^{NB})_B^E}{(u_A^{NB})_B^E} \\ \arctan \frac{(v_A^{NB})_B^E}{\sqrt{(u_A^{NB})_B^E + (w_A^{NB})_B^E}} \end{bmatrix} \]  

(2-246)

\[ \begin{bmatrix} (u_A^{NB})_B^E \\ (v_A^{NB})_B^E \\ (w_A^{NB})_B^E \end{bmatrix} = \left( \begin{array}{c} v_{K}^{G} \end{array} \right)_B^E + \left( \begin{array}{c} \mathbf{\omega}^{IB} \end{array} \right)_B \times \left( \mathbf{r}^{G,NB} \right)_B - \mathbf{M}_{BO} \left( \begin{array}{c} v_{W}^{G} \end{array} \right)_D^E \]  

(2-247)

The matrix \( \mathbf{M}_{BO} \) gives the transformation of NED frame to the Body-fixed frame with the sequence of rotation about \( z \) axis \( \rightarrow \) \( y \) axis \( \rightarrow \) \( x \) axis and with the corresponding rotation angles \( \psi, \theta, \phi \) respectively. Without derivation, the complete matrix is mentioned below.
Furthermore, simple error models consisting of offsets \( \Delta p_{\text{static}}, \Delta p_{\text{dynamic}}, \Delta \alpha, \Delta \beta, \Delta T_{\text{static}} \) and scale factors \([K_{p_{\text{stat}}}, K_{p_{\text{dyn}}}, K_\alpha, K_\beta, K_{T_{\text{stat}}}]\) can be introduced for the outputs \([p_{\text{stat.meas}}, p_{\text{dyn.meas}}, V_{TAS,\text{meas}}, \alpha_{A,\text{meas}}, \beta_{A,\text{meas}}, T_{\text{stat.meas}}]\) along with the corresponding white measurement noise. Whereas, error models of GPS based velocity and position measurements, contain only the white measurement noise. But as previously mentioned, all the bias parameters and scale factors were kept to be zero as the data was generated using ‘X-Plane’ without any offset and scale factor error. Although, these parameters may play an important role for a real flight data, where in reality all the measurements are not synchronous and have some delays.

\[
\begin{align*}
(r^G)_{WGS84,\text{GPS,meas}} &= (r^G)_{WGS84} + v_{r_{\text{GPS}}} \\
(v^G_K)_{0,\text{GPS,meas}} &= (v^G_K)_{0,\text{GPS}} + v_{v_{\text{GPS}}} \\
p_{\text{stat.meas}} &= (K_{p_{\text{stat}}} + 1)(p_{\text{static}} - \Delta p_{\text{static}}) + v_{p_{\text{stat}}} \\
p_{\text{dyn.meas}} &= (K_{p_{\text{dyn}}} + 1)(p_{\text{dynamic}} - \Delta p_{\text{dynamic}}) + v_{p_{\text{dyn}}} \\
\alpha_{A,\text{meas}} &= (\alpha_{A_{\text{meas}}} - \Delta \alpha) \cdot (K_\alpha + 1) + v_\alpha \\
\beta_{A,\text{meas}} &= (\beta_{A_{\text{meas}}} - \Delta \beta) \cdot (K_\beta + 1) + v_\beta \\
T_{\text{stat.meas}} &= (K_{T_{\text{stat}}} + 1)(T_{\text{static}} - \Delta T_{\text{static}}) + v_{T_{\text{stat}}}
\end{align*}
\]

Therefore, the final measurement vector is,

\[
z = \begin{bmatrix}
(r^G)_{WGS84} \\
(v^G_K)_{0,\text{GPS}} \\
p_{\text{stat}} \\
p_{\text{dyn}} \\
\alpha \\
\beta \\
T_{\text{stat}}
\end{bmatrix} + \begin{bmatrix}
v_{r_{\text{GPS}}} \\
v_{v_{\text{GPS}}} \\
v_{p_{\text{stat}}} \\
v_{p_{\text{dyn}}} \\
v_\alpha \\
v_\beta \\
v_{T_{\text{stat}}}
\end{bmatrix}
\]
To summaries, the system’s state and input vectors are

\[
x = \begin{bmatrix}
    (r^G)_{WGS84} \\
    (v^E_K)_B \\
    \psi_{euler} \\
    (v^W_K)_B
\end{bmatrix}
\]

(2-257)

\[
u_{meas} = \begin{bmatrix}
    (a^H_{INS,meas})_{INS,meas} \\
    (\omega^B_{INS,meas})_{INS,meas} \\
    (\dot{\omega}^B_{INS,meas})_{INS,meas}
\end{bmatrix}
\]

(2-258)

Also, the output and measurement equations are

\[
y = h(x, u, w, v) = \begin{bmatrix}
    (r^G)_{WGS84} \\
    (v_{K, GPS})^E \\
    (v^E_K)_{O, GPS} \\
    p_{stat} \\
    p_{dyn} \\
    \alpha \\
    \beta \\
    T_{stat}
\end{bmatrix}
\]

(2-259)

\[
z = \begin{bmatrix}
    \begin{bmatrix}
        (r^G)_{WGS84, GPS, meas} \\
        (v_{K, GPS})^E_{O, GPS, meas} \\
        p_{stat, meas} \\
        p_{dyn, meas} \\
        \alpha_{A, meas} \\
        \beta_{ler, meas} \\
        T_{stat, meas}
    \end{bmatrix} \\
    v_{r, GPS} \\
    v_{v, GPS} \\
    v_{p, stat} \\
    v_{p, dyn} \\
    v_{\alpha} \\
    v_{\beta} \\
    v_{T_{stat}}
\end{bmatrix} = y + \begin{bmatrix}
    v_{r, GPS} \\
    v_{v, GPS} \\
    v_{p, stat} \\
    v_{p, dyn} \\
    v_{\alpha} \\
    v_{\beta} \\
    v_{T_{stat}}
\end{bmatrix}
\]

(2-260)
2.6 Re-entry Vehicle Example

In the current section, an example problem regarding tracking of a Re-entry vehicle is described. The reason for discussion of this example is the presence of significant nonlinearities in the process and measurement models; and based on the extensive analysis done in past, substantial difference was witnessed in the state estimation by extended Kalman filter and unscented Kalman filter, as also shown in (Simon Julier, 2000). The implementation of the problem is presented in section 4.3.

2.6.1 Problem description

The goal in this problem is to estimate the states of a body as it re-enters the earth’s atmosphere, at a very high altitude and with high velocity. The state vector is composed of position \( x_1(t) \), velocity \( x_2(t) \) and constant ballistic coefficient \( x_3(t) \). Furthermore, its motion is determined by altitude and velocity dependent drag terms, and it is constrained to fall vertically.

The continuous time dynamics of the system is given as

\[
\begin{align*}
\dot{x}_1 &= -x_2(t) + w_1(t) \\
\dot{x}_2 &= -e_1^{rx_1(t)} x_2(t)^2 x_3(t) + w_2(t) \\
\dot{x}_3 &= w_3(t)
\end{align*}
\]  

(2-261)

Where, \( \gamma \) is a constant that relates the air density with altitude.

The Measurement equation is

\[
z_1 = \sqrt{(M^2 + [x_1(t) - a]^2)} + v_1(t)
\]  

(2-262)

Where, \( M \) represents the Radar height and \( a \) signifies the horizontal range between the body and radar.

Also, \( Q \) and \( R \) are the process noise covariance matrix and measurement noise covariance matrix respectively and are defined as,

\[
w \approx (0, Q)
\]

\[
v \approx (0, R)
\]  

(2-263)

\[
E[wv^T] = 0
\]

2.6.2 Results

The plots obtained after the filter run are presented. Firstly, both the filters were run once to obtain the estimated states and reconstructed outputs, which are displayed in Figure 2-2 and Figure 2-3, along with the respective true states and measurements. Also the absolute value of the estimation errors, for both UKF and EKF, are shown in Figure 2-4.
The presented results were obtained using the original Matlab scripts (‘m-files’), so the respective computational time (in terms of the CPU time) used and the resultant mean-square error of estimation are listed in Table 2-1 and Table 2-2 respectively.

Figure 2-2: Estimated States

Figure 2-3: Reconstructed Output

Figure 2-4: Absolute estimation error
<table>
<thead>
<tr>
<th></th>
<th>UKF</th>
<th>EKF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Altitude [ft]</td>
<td>0.6242e+05</td>
<td>1.6765e+10</td>
</tr>
<tr>
<td>Velocity [ft/sec]</td>
<td>1.2817e+05</td>
<td>0.0111e+10</td>
</tr>
<tr>
<td>Const. ballistic coefficient</td>
<td>0.0088</td>
<td>0.0015</td>
</tr>
</tbody>
</table>

Table 2-1: Computational time [sec]

Table 2-2: Mean-square estimation error

It is illustrated from Table 2-1, that the time taken by both the filters for the present application is nearly same for both the filters. The reason for the evident result is the specialized UKF algorithm (that assumes the additive nature of the uncertainties), which was used for the system and is a lot faster than the generalized UKF algorithm. However, the time taken by EKF was expected to be small, as for the present system also, the Jacobians are evaluated analytically using MuPAD, instead of the numerical computation (finite difference technique) and is believed to be slower than the former.

For a thorough comparison, the absolute estimation errors for all the states, committed by each filter, were obtained across a Monte Carlo simulation consisting of 50 runs and are presented in Figure 2-5. It is evident in the mentioned figure that, although the absolute estimation error, obtained out of a single run of the filters, illustrates no substantial difference, but then taking average over 50 Monte Carlo simulations, the estimation error in EKF perceived to be far more than the estimation error of UKF.

To explore the linearization robustness of EKF and UKF, the system’s differential equations were integrated over various time steps and the corresponding estimated states and reconstructed output were obtained along with the absolute estimation errors. Thereafter, the absolute estimation errors were averaged over 10 Monte Carlo simulations and are displayed in Figure 2-6. In the mentioned figure, the title over each column, indicates the linearization time step, used for the corresponding estimation process.
Furthermore, the mean-square errors were also obtained for all the states and by each filter, which were later averaged over 10 Monte Carlo simulations and are shown in Figure 2-7; with the 'x-axis' signifying the linearization time step and the 'y-axis' depicting the average mean-square error of the state (in log scale) and its name is given as the title over each plot. As witnessed from the figure, the linearization error in EKF proves to be devastating to its estimation accuracy, as the average mean-square error in EKF increases drastically with the time step more than 0.5 sec. However, the average mean-square error, in case UKF, remains low throughout the variation domain.

As evident from the presented figures, both the filters obtained similar results till 10 secs of simulation time, when the body was at high altitude with minimal drag effects and was approximately falling linearly. Thereafter, the drag increased significantly, which resulted in a highly nonlinear motion and eventually lead both the filters to produce significantly different estimations. Figure 2-5 illustrates that, even in the presence substantial nonlinearities, UKF estimates altitude and velocity with better accuracy than EKF, whereas the estimate of ballistic coefficient is nearly same by both of them. Moreover, the linearization assumptions for EKF broke down at time step of 0.5 sec and consequently, a detrimental estimation performance was established with respect to the estimation performance of UKF, which still obtained satisfactory results.
3 Results

A detailed comparison of UKF and EKF algorithms, for the nonlinear continuous-discrete time reference model, is presented in this chapter. Various basis for comparison are

I. Computational Time.
II. Poor initial guess.
III. Un-tuned noise covariance matrices ($Q$ and $R$).
IV. Lower measuring rates (measurements are obtained at lower frequency).
V. Lower sampling rates (Linearization of the system over larger time step).

On account of the thorough comparison, the main goal is to address two research questions, which are

Q-1. Under what circumstances, both UKF and EKF obtain similar or comparable results?
Q-2. In what all situations, EKF, being a linearization based filter, leads to unrealistic or highly inaccurate state estimation?

The flight data was generated using ‘Simulink’ (a simulation platform in Matlab) and ‘X-Plane’ (a flight simulation software). The purpose of using X-Plane is to generate flight measurements, using the dynamic models of various aircrafts (present in the software) and which are later extracted from X-Plane using a simulation model, made in Simulink. That is, X-Plane is acting like a black box, which provides all the model equations, necessary to generate true states and measurements upon giving inputs, which are then stored and evaluated using Simulink. Amongst various aircraft models present in X-Plane, all the data for current section was obtained using the ‘B-777’ aircraft’s model.

On account of the above mentioned simulation model, true states, true outputs, erroneous states and erroneous measurements along with the given inputs (including noise), were obtained and provided as a startup data to the filter. Also, the initial state vector was computed by taking the average over first few samples of the true states.

To obtain results at the design condition (with known initial states and well-tuned covariance matrices), the differential equations of the system were integrated over a time update of 0.01 sec (the linearization time step). Two sensors were available with different measurement frequencies for the measurements. One was the GPS (Global Positioning System) with measurement rate of 10 Hz and other was the ADS (Air Data System) with 50 Hz measuring frequency. In all the cases, the available measurements were the combined measurements of the GPS and ADS sensors.

The parameter used to check accuracy for comparison of both the filters over all the bases, is the ‘mean-square error’ and is defined as

$$MSE = \frac{1}{N} \sum_{i=1}^{N} e_i^2$$  \hspace{1cm} (3-1)
Comparison of EKF and UKF for Flight Path Reconstruction in System Identification
Mohit Mehdintra

3 Results

Where, \( e_i \) is the estimation error (difference between the true states and the estimated states) in the \( i^{th} \) sample and \( N \) represents the total number of samples.

For all the cases presented in this section, the identical inputs (for instance, initial conditions and covariance matrices) were given to both the filter and the results of two objects of the FPR class (FPR_EKF_Obj and FPR_UKF_Obj) are plotted, as mentioned in section 4.1.1. Overall, five different Maneuvers were performed by the aircraft (in both longitudinal and lateral plane) using X-Plane and the inputs received by the dynamic system are shown in Figure 3-1 to Figure 3-3.
3 Results

The estimated position coordinates along with the flight altitude, are presented in Figure 3-4. As evident from Figure 3-5 that shows the absolute error in position estimation, the longitude ($\lambda^G_{WSB4}$) and lateral ($\Phi^G_{WSB4}$) position coordinates are estimated, equally well, by both of the filters with the difference lying only at third and fourth decimal places respectively i.e., at $10^{-3}$ deg and $10^{-4}$ deg. On the other hand, in case of the altitude (last plot in the same figure), the EKF starts diverging after 5 sec, whereas the error in UKF remains low. Still the estimation error is found to be less than 1 m for both of them, which is 'good' as the plane was flying at approximately 5000 m altitude.
The estimated kinematic velocities are shown in Figure 3-6 followed by their absolute estimation errors in Figure 3-7. All the kinematic velocity components are equally estimated by EKF and UKF, as plots of both the filters almost overlap each other and one can hardly distinguish any distinct plot in both the figures (Figure 3-6 and Figure 3-7). Also, it can be seen in Figure 3-7 that, the 'x' component ($u^G_k$) of the kinematic velocity is best estimated by the two, compared to the other components, as the error is less than $0.02 \text{ m/s}$ for all the maneuvers with an approximate flight speed of $200 \text{ m/sec}$ (along ‘x’ direction). The reason for the 'not so good' estimation of ‘y’ and ‘z’ components (by both the filters) are the high wind velocities along ‘y’ and ‘z’ directions ($v^W_k$ and $w^W_k$) during all the maneuvers, which can be seen in Figure 3-10 and are approximately equal to the flight velocity components ($v^G_k$ and $w^G_k$) in those directions.
Comparison of EKF and UKF for Flight Path Reconstruction in System Identification

The estimation for the attitude angles, shown in Figure 3-8, is almost equal by EKF and UKF, as both the plots in Figure 3-8 and Figure 3-9 are overlapping and shows no considerable difference. The estimation is ‘good’ for these states too, since the estimation error in the whole flight duration and for all the maneuver did not even reach 0.5 $\text{deg}$, considering the influence of the unit ($\text{deg}$ in this case) over the huge structure of the aircraft.

The estimation is ‘poor’ for the last three states of the state vector, which are the wind components, compared to all the other states but the plots of both the filters are again overlapping and show no substantial difference, as can be seen in Figure 3-10 and Figure 3-11. Although the wind velocity components are not correctly estimated and estimation error is almost equal to the magnitude of the actual wind components but as an achievement for both
3 Results

The reconstructed outputs are shown in Figure 3-12 to Figure 3-14 along with the actual measurements. As can be seen in the figures, all the outputs (positions, velocities and air data quantities) are equally well tracked by both the filters, with the output always at the center of the noisy measurements. Here also, the plots of UKF and EKF almost overlap each other and UKF did not show any considerable improvement over EKF.
3 Results

Comparison of EKF and UKF for Flight Path Reconstruction in System Identification

Mohit Mehndiratta

All the results presented above were obtained using the filters only. Further, upon using the smoothers, the accuracy of smoothed states was found to be only marginally different from the estimated states and hence, the corresponding results are not presented; instead the figures are shown in appendix B.1.

Based on all the results presented till now, it was observed that in the case of well-tuned parameters (covariance matrices) and design or optimum conditions (accurately known initial states and error covariance matrix, high measurement frequency and low linearization time step), UKF did not show any specific improvement over EKF, except in estimating the position altitude (Figure 3-5).
Comparison of EKF and UKF for Flight Path Reconstruction in System Identification
Mohit Mehndiratta

Figure 3-13: Reconstructed Outputs – Kinematic velocities

Figure 3-14: Reconstructed Outputs – Air data sensor quantities
Therefore, for further comparison of UKF and EKF, based on the factors mentioned at the starting of the current chapter, off-design conditions were focused and the results obtained are presented in the following sections. One can note at this point, that for all the following discussions, the filtered estimates were considered instead of the smoothed estimates due to lack of improvements observed in the estimates using the smoothers, as stated already; also, the main focus of the report lies in the comparison of the filters rather than their smoothers.

### 3.1 Computational Time

In the present section, the goal is to weigh the performance of both the filters (EKF and UKF) with respect to the computational complexity. That is, the achieved estimation accuracy by both of them is compared with the computational time used. For this, the corresponding computational time (in terms of ‘CPU time’ in [sec]) taken by ‘m-file’ (Matlab script) and by ‘mex-file’ (compiled script), for all the five maneuvers (same as used previously), are listed in Table 3-1. It can be seen in the table that the purpose of compilation of ‘m-files’ to ‘mex-file’ (as mentioned in Chapter 4) is justified as the reduction in the computation time, especially for UKF, is almost 98% and is more than significant.

In case of the table for mean-square error, average was taken over all the five maneuvers and therefore, average mean-square error for each state is listed in Table 3-2. This was done on account of the similar order of magnitude of the mean-square error obtained in case of each state and for all the maneuvers, as also seen in the figures of the previous section. Apparently, with a closer look at the Table 3-2 one can realize that except for the average mean-square error of the position altitude \((h^g)_{WGS84}\) for which both the filters are showing different results (as stated previously also), in case of the average \(MSE\) for all the other states, same order of accuracy was obtained. Furthermore, it was observed that for some states EKF has lower \(MSE\) and for some other states UKF has lower \(MSE\).

<table>
<thead>
<tr>
<th>Maneuver</th>
<th>UKF ‘m-file’</th>
<th>UKF ‘mex-file’</th>
<th>EKF ‘m-file’</th>
<th>EKF ‘mex-file’</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>104.4605</td>
<td>2.4597</td>
<td>2.5597</td>
<td>0.2779</td>
</tr>
<tr>
<td>2</td>
<td>105.6457</td>
<td>2.3675</td>
<td>2.0685</td>
<td>0.2692</td>
</tr>
<tr>
<td>3</td>
<td>105.2879</td>
<td>2.3978</td>
<td>2.0491</td>
<td>0.2865</td>
</tr>
<tr>
<td>4</td>
<td>106.521</td>
<td>2.372</td>
<td>2.0881</td>
<td>0.2732</td>
</tr>
<tr>
<td>5</td>
<td>105.7468</td>
<td>2.3834</td>
<td>2.6905</td>
<td>0.2688</td>
</tr>
<tr>
<td>Average</td>
<td>105.5324</td>
<td>2.3961</td>
<td>2.2912</td>
<td>0.2751</td>
</tr>
</tbody>
</table>

**Table 3-1: Computational time [sec]**

<table>
<thead>
<tr>
<th>State</th>
<th>EKF</th>
<th>UKF</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\lambda^g)<em>{WGS84})(</em>{GPS}) ((deg))</td>
<td>3.693e-08</td>
<td>4.457e-08</td>
</tr>
<tr>
<td>((\phi^g)<em>{WGS84})(</em>{GPS}) ((deg))</td>
<td>2.549e-08</td>
<td>2.396e-08</td>
</tr>
<tr>
<td>((h^g)<em>{WGS84})(</em>{GPS}) ((m))</td>
<td>0.15069</td>
<td>0.03179</td>
</tr>
<tr>
<td>((u^g)_{INSIMU}^E) ((m/s))</td>
<td>3.728e-05</td>
<td>3.667e-05</td>
</tr>
<tr>
<td>((v^g)_{INSIMU}^E) ((m/s))</td>
<td>0.15168</td>
<td>0.15872</td>
</tr>
<tr>
<td>((w^g)_{INSIMU}^E) ((m/s))</td>
<td>0.03887</td>
<td>0.04205</td>
</tr>
</tbody>
</table>
### Results

| \( \phi_{IMU} \) [rad] | \( 5.069 e - 06 \) | \( 5.189 e - 06 \) |
| \( \theta_{IMU} \) [rad] | \( 1.163 e - 06 \) | \( 1.252 e - 06 \) |
| \( \psi_{IMU} \) [rad] | \( 3.353 e - 06 \) | \( 3.505 e - 06 \) |
| \( (u^E_{W0})_O \) [m/s] | \( 0.58749 \) | \( 0.5949 \) |
| \( (v^E_{W0})_O \) [m/s] | \( 0.130366 \) | \( 0.130385 \) |
| \( (w^E_{W0})_O \) [m/s] | \( 0.48316 \) | \( 0.48651 \) |

**Table 3-2: Average mean-square estimation error**

On comparing the results of the Table 3-1 and Table 3-2, it can be seen that to obtain similar order of accuracy, UKF always takes the computational time which is 8 – 9 times of the time taken by EKF using ‘mex-file’ (also, UKF takes 40 – 50 times of the time taken by EKF for ‘m-file’) and hence, EKF outperforms UKF in case of the computational burden caused. The reason being the increased number of computations that occur in the general case of UKF (used here), which has an augmented form of the state vector and thereafter, results in an increased number of sigma points evaluation during each propagation step. Whereas, in case of the EKF, only the additional Jacobian matrices are to be evaluated during each propagation, which is very fast for the present case, as instead of computing them (Jacobians) numerically (using finite differences), analytical expressions were obtained with separate Matlab scripts, which were created for each one of them (Jacobian) using MuPAD as, mentioned in Chapter 4.

#### 3.2 Poor Initial Guess

As mentioned earlier, at the *design* condition, the initial estimates for the states were obtained by taking average over few starting samples. To study the performance of both the filters at off-design conditions, one of the chosen parameters is the ‘poor’ guess (or ‘crude’ approximation) for the initial state vector, which was obtained by varying the used initial states (given as an input data to the filters) in percentage of the true initial states and with respect to the true initial states themselves (known from X-Plane). That is, various starting state vectors were obtained by percentage addition (and subtraction) of the true initial states on both sides of the true initial state vector (keeping all other parameters at their optimum values) and were used for the estimation process afterwards, together with the computation of the respective estimation errors. Thereafter, the mean-square error with varying initial conditions for each state was calculated and an average was taken over the similar five maneuvers as used in the previous section.

The aim of current section is not to estimate the accuracy achieved, but to compare the performance of the two filters, based on the convergence of the state estimates to their true values, for the degraded initial conditions. The main interest is in obtaining the time taken to converge and the limit of variation for convergence that is, the maximum percentage variation UKF estimation can still withstand, but for which estimates of EKF diverges from the true values.

Therefore, for a precise performance comparison between the two filters, all the estimated states and reconstructed outputs are shown only for the first maneuver (out of the five), since a similar behavior was observed in case of the other maneuvers as well. Also, the results are shown for the maximum variation of ±25 %, as beyond this limit, a divergence was observed.
Results

Comparison of EKF and UKF for Flight Path Reconstruction in System Identification

Figure 3-15: States – Position with varying initial conditions

Figure 3-16: States – Kinematic Velocities with varying Initial conditions

for most of the estimated states by both the filters. The mentioned results are given in Figure 3-15 to Figure 3-21, where the title over each column represents the percentage variation from the true values.

As the previous data was reused, it can be seen that in some of the cases (mostly for extreme variations), the complete convergence was not obtained within the given sampling time, hence the corresponding convergence time by both EKF and UKF cannot be known for each state. But nevertheless, by looking at the pattern of the change of a state, the convergence behavior of both the filters can be observed and conclusions can be drawn as to which filter converges faster.
From all the mentioned figures, it can be perceived that for a small variation of ±5% both of the filters almost converge simultaneously, except for some states like position altitude \( (h^G)_{WGS84} \), in which EKF did not converge in the given sampling time whereas, UKF converged very fast and apparently for some other states (like attitude angles in Figure 3-17 and wind velocities in Figure 3-18) even for this small variation (±5%), EKF diverged. A similar behavior can also be seen for the reconstructed outputs i.e., either the reconstructed outputs of EKF diverge from the measurements or their convergence rate is lower than that of the UKF.
3 Results

The probable reason for EKF to show a slower convergence to the true values is the ‘Linearization’ principle involved in the ‘prediction step’ of the estimation process. That is, in EKF the state covariance matrix (which accounts for the uncertainties in the states) is propagated in a linearized manner, leading to the presence of an inherent linearization error; whereas UKF propagates the state covariance matrix in a nonlinear way and does not have any linearization error. Therefore, it seems that the linearization error in EKF contributes a major role towards its slower convergence as compared to UKF.

In all the above mentioned cases, the initial state covariance matrix was kept equal to the value, as was used in the design condition. When, even the initial covariance matrix was changed to obtain results for all the various initial conditions (as used above), a behavior
Comparison of EKF and UKF for Flight Path Reconstruction in System Identification
Mohit Mehndiratta

Figure 3-21: Outputs – Air Data quantities with varying Initial conditions

Figure 3-22: Average MSE for percentage variation in Initial conditions
different from the expected was observed. That is, from intuition it was believed that in case of highly erroneous initial estimates, an increase in the initial covariance matrix would lead to a faster convergence for both the filters along with an increase of the maximum allowable limit of variation, before divergence. But apparently, for increase of the initial covariance matrix, the accuracy of both the filters was reduced and a faster convergence was only achieved for UKF whereas EKF diverged even sooner. The results obtained are presented in appendix B.2 (Figure B-12 to Figure B-19), due to their less significance regarding the concerned parameter.

Again the respective mean-square errors for each state, averaged over five maneuvers are shown in Figure 3-22 and for further comparison, the variation limit is extended to $\pm 100\%$. Also, from the current figure, It is evident that for each state the mean-square estimation error caused by EKF is always higher than the mean-square estimation error made by UKF, for a particular percentage variation and moreover, EKF always shows a lower rate of decrease of the $MSE$ than the rate of decrease of $MSE$ as shown by UKF, with decreasing percentage variation i.e., for improvement in accuracy of the initial states.

From all the above discussion, it can be concluded that for small variations, both the filters show similar estimation performance and accuracy, which gets degraded as the error in the initial state vector increases. But for the comparison of the filters, UKF attains the convergence first, for all the cases with larger percentage of initial errors. Therefore, UKF is said to be more robust and superior, regarding the poor initial estimates of the state vector, with respect to EKF.

### 3.3 Un-tuned Noise Covariance Matrices

As already known from theory that noise covariance matrices give a measure of ‘trust’ for the sensor measurements with the higher covariance indicating a lower trusted measurement. Moreover, due to the generation of data using ‘X-Plane’, good approximations for the process noise covariance matrix $Q$ and measurement noise covariance matrix $R$ were available for the present case. But apparently for a real flight test, approximate values of these noise covariance matrices can be obtained either by laboratory experiments or by a Global Fourier Smoothing technique, which is based on the principle of transforming a signal into the frequency domain and thereafter, ignoring all the frequencies higher than a certain threshold, followed by its retransformation to time domain. These obtained approximations were later improved by a ‘trial and error’ method; such that to obtain a best fit between the estimated states and the true states, along with a good overlapping of the measurements on the reconstructed outputs. Hence, the optimum value of the noise covariance matrices ($Q$ and $R$) for the particular case (the dynamic system considered in the report) is known. To study the robustness of both the filters at off-design condition regarding this parameter, that is with respect to the varying noise covariance matrices; both $Q$ and $R$ matrices were varied individually by the order of magnitude of $10$ and the estimation was done for the five maneuvers (same as earlier) to obtain the corresponding mean-square errors.
Firstly, the individual variation of $Q$ matrix is performed keeping $R$ constant and the estimated states and reconstructed outputs are presented in Figure 3-23 to Figure 3-29. In the displayed figures, a substantial effect was only observed for the extreme variations on both the sides (positive and negative variations) and therefore for a concise representation, the estimated states and reconstructed outputs are plotted over the first maneuver (out of the five maneuvers as used previously) and for the extreme cases only. The title over each column of a plot represents the variation by that ‘exponential power’ of 10 and the axes represent the change of a state (or output) over time.
It is evident in the shown figures that for most of the states and outputs, the estimation by EKF portrays better results than UKF over the given variations of the noise covariance matrices. For instance, in the kinematic velocity estimation of Figure 3-24, the results of UKF displayed lot of fluctuations and beside that, a large offset is also witnessed from the true values for positive variations of the noise covariance matrix as compared to the results of EKF which are closer to the true values. Similar behavior can be observed in the case of other states.
A good overlapping was established by the reconstructed position outputs of EKF and UKF, for both the negative and positive variations of $Q$ matrix, as depicted in Figure 3-27. But then in case of most of the other outputs, the reconstruction by EKF still tracks the measurements, however the reconstructed outputs by UKF are well off for the positive variations in $Q$ matrix.
3 Results

Comparison of EKF and UKF for Flight Path Reconstruction in System Identification

Figure 3-29: Outputs – Air Data quantities with varying $Q$

Figure 3-30: Average MSE for Variation in $Q$
Now, to compare the estimation error occurred due to the variation of $Q$ matrix from its optimal value, the mean-square error in each state averaged over the five previously used maneuvers, is shown in Figure 3-30. In the displayed figure, the ‘y-axis’ in each plot signifies the average $MSE$ for the mentioned state whereas the ‘x-axis’ depicts the variation in a particular noise covariance matrix from its known optimum value, and the name of the covariance matrix is mentioned as the ‘x-label’ of the last subplots. For the variation of $Q$ in particular, it can be witnessed that for most of the states in Figure 3-30, the average mean-square error remains nearly constant, as obtained by both the filters for a negative variation (or with negative powers of 10) of the covariance matrix and thereafter, it gradually increases along the positive side. Only in the case of position altitude ($h^G_{WGS84}$) where UKF dominates the accuracy, but for the rest of the states, performance of both is either same or EKF appears to be better.

On the other hand, for an individual variation of $R$ matrix, not much difference was perceived from the plots, as already obtained for the only variation in $Q$ matrix. Therefore, instead of presenting all the figures for estimated states and reconstructed outputs again in the current section, they are given in Appendix B.3 (Figure B-20 to Figure B-26); But for an efficient comparison of the filter performance for the only variation in $R$ matrix, the mean-square errors in all the states, averaged over five maneuvers, are illustrated in Figure 3-31.

One can note from Figure 3-30 and Figure 3-31 that, in majority of the plots for both the variation cases of the noise covariance matrices, the average $MSE$ is minimum at the center that is, when the variation is zero and hence, justifies the argument that the optimum values for $Q$ and $R$ matrices are known.

In case of the individual variation of $R$ matrix, shown in Figure 3-31, the extreme negative powers indicate superiority in performance by UKF, as for majority of the states EKF has higher average $MSE$, which however decreases with UKF till the center. Furthermore, along the positive variation side, EKF has lower average mean-square error only for the position states ($\lambda^G_{WGS84}$, $\phi^G_{WGS84}$ and $h^G_{WGS84}$) but, the average mean-square error by both the filters for the rest of the states, is nearly the same.
The reason for obtaining a ‘poor’ performance at the extreme variations by UKF is that in the general form of UKF (with augmented state vector), the uncertainties enter the nonlinear system directly, through the process noise and measurement noise sigma vectors ($\chi^k_{m-1}$ and $\chi^k_{m-1}$). So if the noise covariance matrices are too small, the sigma vectors will be concentrated near the mean and will not be able to capture the uncertainties which lie outside the cloud (or area) of the sigma points. On the other hand, if the magnitude of the noise covariance matrices are too large, the sigma points will be far away from the mean and the low magnitude uncertainties will be left unaccounted.

However, in the case of EKF the uncertainties indirectly enter the state estimation process through the Kalman Gain matrix ($K_k$), which itself is a weighing measure and compares the accuracy of the predictions (predicted states and covariance matrix) with the measurements, before computing the corrected states. Consequently, for higher magnitude of the process noise covariance matrix $Q$, the measurements are more trusted, that even leads to ‘bad’ estimation for some of the states, as can be seen in Figure 3-26; whereas, for higher measurement noise covariance matrix $R$, the evaluation of the corrected states are more influenced by the magnitude of the predicted (or uncorrected) estimates than the measurements.

As concluding from the Figure 3-30 and Figure 3-31, the EKF has proven to be more robust for the un-tuned parameters (noise covariance matrices) over the UKF as for majority of the states, in case of individual variations both $Q$ and $R$ matrices, the average mean-square error by EKF is always lower than that of the UKF, especially for the extreme variations.

### 3.4 Lower Measuring Rates

As mentioned earlier, two sensors GPS and ADS with measuring rates of 10 Hz and 50 Hz respectively, were used simultaneously to obtain the measurements. The aim of current section is to study the effect of changing measurement frequencies over the estimation process and compare the respective estimation errors obtained by both the filters. Again the same five maneuvers were used to perform the estimation, with the decreasing measurement sampling rates and the corresponding mean-square errors were obtained.

Even though the figures for all the estimated states and reconstructed outputs with different measurement rates were available, for all the maneuvers, only the mean-square errors averaged over the five maneuvers are illustrated for conciseness since the results of both the filters were almost overlapping and no additional information could be drawn from those plots. It is observed from Figure 3-32 that the average $MSE$ for most of the states obtained by both the filters, decrease nearly in the same manner and by the similar amounts with the increasing measurement frequency except for the position altitude ($h_G$)$_{WGS84}$, where the average $MSE$ by EKF remains constant along the whole frequency domain.

As evident from Figure 3-32, the measurement frequency does not show a high influence over the estimation accuracy of both the filters. The observation made is contradicting to what was expected, as from intuition it was thought that at low measuring rates, the estimation error of EKF should always be higher than the estimation error caused by UKF. This is because of the presence of the linearization error in the propagation of the state error covariance matrix in case of EKF while UKF is free from any of these linearization error and hence, should result in a better performance at the lower measuring rates.
Moreover, with the increase of the measurement rate the estimation error should decrease, especially in case of EKF, as higher number of measurements would be available to compensate for the linearization errors.

A possible reason for observing the above mentioned behavior would be the presence of a modeling error in the system’s dynamic equations. On account of this error, the system’s model appears to be perfectly known to the filter. Consequently, it results in the minimum influence of measurements over the state estimation and beside that, even accurate enough estimates are obtained without measurements.

As concluding remarks, both EKF and UKF perform equally well, for the considered dynamic system, with the varying measurement rates. Furthermore, the measuring frequency does not affect the estimation accuracy significantly for both the filters, provided the other parameters are kept equal to their values as that at the design condition; for instance, well known initial states and tuned noise covariance matrices.

### 3.5 Lower Sampling Rates

As understood from Chapter 2, the standard Kalman filter is applicable to linear stochastic systems only and its direct implementation over the nonlinear systems is not possible. To address this problem, one of the artifices is to linearize the system about the most recent state estimates and the technique is known as Extended Kalman filter (EKF). Apparently, this linearization step also forms the major shortcoming in EKF approach, as the linearization time step is need to be sufficiently small to obtain accurate state estimates. Thus, the aim of the current section is to evaluate the influence of lower sampling rates (or higher linearization times) over the performance of both the filters, especially EKF, as UKF does not have any linearization assumption and beside that, to obtain a limit (a kind of ‘threshold’) for the assumption to be valid. This implies in determining the maximum linearization time step, so that the filters do not break and still produce enough accurate estimates. Although UKF does not involve any linearization step, but one can keep this in mind that for the prediction step,
To study the performance of EKF and UKF regarding the present parameter (linearization time), different data set comprising of measurements, inputs, time vector and true states, were generated with varying sample size. The estimated states and reconstructed outputs by both the filters for the first maneuver (out of the previously used five maneuvers) are presented in Figure 3-33 to Figure 3-39, where the title over each column depicts the linearization time step used in the corresponding estimation process. One can note that, even though the sampling time was varied, the measurement rate in each case was kept constant and equal to that at the design condition, which is 50 Hz.
As witnessed from the figures of the estimated states, both the filters estimate the majority of the states equally well and with sufficient accuracy, for the sampling frequency higher than 10 Hz (or \( \Delta t \leq 0.1 \text{ sec} \)); which degrades simultaneously for both, with a further increase of the sampling time. Evidently, for the present case \( \Delta t = 0.1 \text{ sec} \) can be regarded as the ‘threshold limit’, until when the results of both the filters are accurate enough. A similar behavior can also be illustrated in case of the reconstructed outputs by both the filters, which sufficiently tracks the measurements for the linearization time step less than the ‘threshold limit’ (\( \Delta t \leq 0.1 \text{ sec} \)).

When the sampling time greater than the threshold (\( \Delta t > 0.1 \text{ sec} \)) was considered, it was established that for majority of the states, the state estimates by EKF are more close to the...
true values, in comparison to the state estimates by UKF except for attitude angle $\psi_{IMU}$ and the kinematic velocity component $(v_{K}^{E})_{S,IMU}$, in which the UKF estimates are more close to the true values, as evident from Figure 3-34 and Figure 3-35 respectively. Another measure of comparison, that is the average mean-square error over five maneuvers, is also displayed in Figure 3-40. Upon a close look one can realize that the illustrated figure also depicts a pattern similar to the previous figures. That is, for the sampling time of greater than the threshold limit, majority of estimates by EKF show either a lower or nearly the same average mean-square error than that displayed by UKF.
Comparison of EKF and UKF for Flight Path Reconstruction in System Identification
Mohit Mehndiratta

Figure 3-39: Outputs – Air Data quantities with various Sampling times

Figure 3-40: Average MSE for Variation in sampling time
However, the obtained behavior is conflicting to what was expected from the results. EKF being a linearization based filter was believed to produce degraded estimates over UKF, with the increase in the linearization time as already seen in the discussion of the example problem of section 2.6 (in Figure 2-6 and Figure 2-7). But on the contrary, as witnessed in Figure 3-40, till the threshold limit of the sampling time, both were producing the similar estimates and a further increase in the sampling time results in the better estimates by EKF rather than UKF.

The plausible reason for the attained estimates might also be the modeling error present in the system dynamic model as mentioned before, in the reasoning of section 3.4. In the present case also, it appears to the filter that the system model is perfectly known, which consequently minimizes the influence of the linearization error in the state estimation and thereafter, resulting in the similar estimates by both EKF and UKF even for larger sampling times.

As concluding remarks, both EKF and UKF perform equally well, for the considered dynamic system, with the varying measurement rates. Furthermore, the measuring frequency does not affect the estimation accuracy significantly for both the filters, provided the other parameters are kept equal to their values as that at the design condition; for instance, well known initial states and tuned noise covariance matrices.

Based on the above discussion the drawn conclusion is that, in contradictory to the anticipated behavior of EKF regarding the decreasing sampling frequency, the estimation accuracy of both the filters, for the sampling time lower than the threshold, is of the same order. Moreover, the magnitude of the estimation error increases for both the filters with a further widening of the linearization time step, but still EKF proves to be more impervious to increase of sampling time than UKF.
4 Documentation

This section describes the software implementation of the two filters. The programming was done using MathWorks software Matlab based on object oriented approach. To achieve better accuracy in computing derivatives (Jacobian matrices), symbolic manipulations were favored but due to extremely slow nature of symbolic calculations in Matlab, another platform of the same software MuPAD was used. All the aircraft’s kinematic equations, as seen in section 2.5.1, were programmed in MuPAD and system matrices, output matrices and their respective Jacobian matrices were obtained in the form of ‘m-files’ (Matlab scripts).

As mentioned before, the basis used was object oriented programming (OOP), hence two main classes named FPR and PROBLEM were developed and their respective objects FPR_Obj and PROBLEM_Obj were created. Inheritance, which is an important property of OOP, was also utilized by simultaneously deriving two sub-classes named UKF and EKF from the main class PROBLEM. A class UML (Unified modeling language) diagram, to give an insight to their structure is mentioned below and thereafter follows the detailed description.

![Figure 4-1: Class UML](image)

Comparison of EKF and UKF for Flight Path Reconstruction in System Identification
Mohit Mehndiratta
4. Detailed Class Description

4.1. The FPR class

This is the one of the main class whose object is created in the main file and its primary purpose is to pass on inputs describing the system, from user to the fellow `PROBLEM` class. The inputs are passed as arguments to the constructor of `FPR` class. Two of its objects are created, one for each filter, `FPR_Obj_UKF` (object that utilizes UKF for filtering) and `FPR_Obj_EKF` (object that utilizes EKF for filtering).

Properties of the class FPR

This class has two sets of properties,

- Properties with `SetAccess` : protected and `GetAccess` : public

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
<th>Type</th>
<th>Dim</th>
</tr>
</thead>
<tbody>
<tr>
<td>choice</td>
<td>‘1’ for choosing UKF and ‘2’ for choosing EKF</td>
<td>double</td>
<td>1x1</td>
</tr>
<tr>
<td>smooth</td>
<td>‘yes’ for obtaining smoothed estimates</td>
<td>string</td>
<td>1x1</td>
</tr>
<tr>
<td>Names_sys</td>
<td>Structure containing input, measurements and states names</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>System_var</td>
<td>Data containing input, measurements and time by user</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>Names_fun</td>
<td>Function names provided as inputs</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>Initial_cond</td>
<td>Starting values for states and covariance matrix</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>x_filter</td>
<td>Reconstructed states obtained from the filter</td>
<td>double</td>
<td>varying</td>
</tr>
<tr>
<td>P_filter</td>
<td>Reconstructed covariance matrix obtained from the filter</td>
<td>double</td>
<td>varying</td>
</tr>
<tr>
<td>y_filter</td>
<td>Reconstructed outputs obtained from the filter</td>
<td>double</td>
<td>varying</td>
</tr>
<tr>
<td>Cov_y_filter</td>
<td>Reconstructed output covariance matrix obtained from the filter</td>
<td>double</td>
<td>varying</td>
</tr>
<tr>
<td>x_filter_s</td>
<td>Smoothed states obtained using smoother</td>
<td>double</td>
<td>varying</td>
</tr>
<tr>
<td>P_filter_s</td>
<td>Smoothed covariance matrix obtained using smoother</td>
<td>double</td>
<td>varying</td>
</tr>
<tr>
<td>y_filter_s</td>
<td>Smoothed outputs obtained using smoother</td>
<td>double</td>
<td>varying</td>
</tr>
</tbody>
</table>

Table 4-1: Properties of class FPR with protected access

- Properties with `SetAccess` : public and `GetAccess` : public

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
<th>Type</th>
<th>Dim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noise_cov</td>
<td>Structure containing noise covariance matrices</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>params_struct</td>
<td>Flight data as input by user</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>compile_flag</td>
<td>‘yes’ to use compiled mex-files</td>
<td>string</td>
<td>1x1</td>
</tr>
<tr>
<td>MSE</td>
<td>Stores the mean-square error value in state estimation, over entire run time.</td>
<td>double</td>
<td>varying</td>
</tr>
</tbody>
</table>

Table 4-2: Properties of class FPR with public access
Methods of the class FPR

The constructor of this class is invoked as,

\[
FPR\_Obj = FPR \( \text{choice}, \text{smooth}, \text{Names\_sys}, \text{System\_var}, \text{Names\_fun}, \ldots \text{params\_struct}, \text{Initial\_cond} \)
\]

Beside constructor invoking which is a static method, other dynamic methods were also used and are listed in Table 4-3

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>calling ( FPR_Obj)</td>
<td>Creation of PROBLEM_Obj (object of PROBLEM class) using input argument, which is called to define the nonlinear system. The values of properties choice and smooth are obtained from FPR_Obj, which subsequently lets it decide which filter to use and whether or not to obtain smooth estimates respectively.</td>
</tr>
<tr>
<td>compile_flag = compile ( FPR_Obj, max_samples, path)</td>
<td>It does the compiling of m-files using ‘Codegen’, an application in Matlab which generates mex-files out of the former and considerably reduces the run time. It returns the value of the property compile_flag as ‘true’ whenever called.</td>
</tr>
<tr>
<td>plot_results ( FPR_Obj, max_fig, … x_errorfree)</td>
<td>It takes FPR_Obj and x_errorfree (true states) as input arguments to plot the reconstructed states, outputs and the estimation error of the filter. Input max_fig gives the number of subplots per figure.</td>
</tr>
<tr>
<td>plot_results_compare2 ( FPR_Obj_UKF, FPR_Obj_EKF, max_fig, … x_errorfree)</td>
<td>This method takes two objects FPR_Obj_UKF and FPR_Obj_EKF as input arguments beside x_perfect and whenever called, it simultaneously plots the reconstructed states and outputs of both the filters and also the estimation errors made by both.</td>
</tr>
<tr>
<td>MSE = calculate_MSE ( FPR_Obj, x_perfect)</td>
<td>It calculates the mean-square error in the state estimation over the entire run time.</td>
</tr>
</tbody>
</table>

Table 4-3: Methods of class FPR
4.1.2 The PROBLEM class

The purpose of this class is to define the system and initialize the predicted state vector and covariance matrix using inputs from the user. The object of this class is named as PROBLEM_Obj and is created in the FPR class. Being the base class, it inherits its properties to two of its sub-classes UKF and EKF.

Properties of the class PROBLEM

It only has properties with SetAccess : protected and GetAccess : public and they are listed in Table 4-4

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
<th>Type</th>
<th>Dim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Names_struct</td>
<td>Structure containing input, measurements and states names</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>Inp_struct</td>
<td>Structure containing values of inputs, time steps, time, measurements and number of measurements</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>Dimen</td>
<td>Stores the dimensions of inputs, states, outputs, process noise, measurements noise and augmented states.</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>Noise_struct</td>
<td>Structure containing noise covariance matrices</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>params_struct</td>
<td>Flight data as input by user</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>Nonlin_struct</td>
<td>Function names of system and output equations</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>Jacobians_struct</td>
<td>Function names to obtain Jacobian matrices</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>States_uncorr</td>
<td>Structure to store uncorrected states</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>Cov_uncorr</td>
<td>Structure to store uncorrected covariance matrices</td>
<td>struct</td>
<td>1x1</td>
</tr>
</tbody>
</table>

Table 4-4: Properties of class PROBLEM with protected access

Methods of the class PROBLEM

The constructor of this class is created as,

```
PROBLEM_Obj = PROBLEM (Names_sys, System_var, Names_fun, ...
Noise_cov, params_struct, Initial_cond)
```

The following dynamic methods are defined for this class

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>system_def (PROBLEM_Obj)</td>
<td>This function is called to define the system by evaluating dimensions of inputs, states, outputs, process noise and measurement noise and store their values in the property Dimen.</td>
</tr>
<tr>
<td>initialisation (PROBLEM_Obj)</td>
<td>This function stores the initial values of predicted states and covariance, as entered by the user, in States_uncorr and Cov_uncorr respectively.</td>
</tr>
</tbody>
</table>

Table 4-5: Methods of class PROBLEM
4.1.3 The UKF class

This class is derived from `PROBLEM` class, hence it inherits all of the base class properties and methods. This class calls an external function `UKF_fun_aug_unstored`, which implements the working of unscented Kalman filter and produces the reconstructed or smoothed (as desired), states and outputs. The object of this class is named as `UKF_Obj` and is created in `FPR` class.

Properties of the class UKF

Beside the inherited properties, those which are defined in this class, with `SetAccess`: protected and `GetAccess`: public, are listed in the following Table 4-6

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
<th>Type</th>
<th>Dim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter_params</td>
<td>Structure containing the filter parameters like ( \kappa, \alpha ) and ( \beta )</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>States_corr_smooth</td>
<td>Structure to store corrected and smoothed state values</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>Cov_corr_smooth</td>
<td>Structure to store corrected and smoothed covariance matrices</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>Recon_measurements</td>
<td>Structure to store reconstructed outputs</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>Smooth_measurements</td>
<td>Structure to store smoothed outputs</td>
<td>struct</td>
<td>1x1</td>
</tr>
</tbody>
</table>

Table 4-6: Properties of class UKF with protected access

Methods of the class UKF

The constructor of this class is created as,

\[
UKF_Obj = UKF ( PROBLEM_Obj )
\]

The dynamic methods defined for this class are listed in Table 4-7

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial_UKF (UKF_Obj)</td>
<td>This function takes <code>UKF_Obj</code> as an input argument and preallocate memory to all its properties.</td>
</tr>
<tr>
<td>ARGS = Initial_compile_UKF (UKF_Obj, N)</td>
<td>This method takes <code>UKF_Obj</code> and <code>N</code> as inputs and returns a structure <code>UKF_struct</code>, to be used as entry-point function in compiling. Also input <code>N</code> gives the maximum allowable sampling size.</td>
</tr>
<tr>
<td>Compiler_fun_UKF (UKF_Obj, max_number_samples)</td>
<td>Compiling of external scripts <code>UKF_fun_aug_unstored</code> and <code>UKF_Smooth</code> is done by this method, where <code>max_number_samples</code> defines the maximum sampling size allowed during run time.</td>
</tr>
<tr>
<td>[ UKF_Obj ] = UKF_fun_aug_unstored_fct_call ( UKF_Obj, compile_flag, smooth)</td>
<td>Based on the value of <code>compile_flag</code> it calls <code>m-file</code> or <code>mex-file</code> to compute reconstructed states and outputs, by implementing the UK filtering algorithm. Also, the property <code>smooth</code> lets it decide whether or not to compute smoothed estimates.</td>
</tr>
<tr>
<td>[ x, P, y, Cov_y ] = gets_function_val ( UKF_Obj)</td>
<td>It extracts and returns the values of reconstructed states, reconstructed states, and reconstructed outputs.</td>
</tr>
</tbody>
</table>
4.1.4 The EKF class

Most of the features of UKF class and EKF class are similar as they both are the derived classes of the PROBLEM class, thus inherits all of the base class properties and methods. This class also calls an external function EKF_fun, which employs the working of extended Kalman filter and produces the reconstructed or smoothed, states and outputs. The object of this class is named as EKF_Obj and is created in FPR class as well.

Properties of the class UKF

Properties defined for this class other than the inherited ones, with SetAccess : protected and GetAccess : public, are listed in Table 4-8

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
<th>Type</th>
<th>Dim</th>
</tr>
</thead>
<tbody>
<tr>
<td>States_corr_smooth</td>
<td>Structure to store corrected and smoothed state values</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>Cov_corr_smooth</td>
<td>Structure to store corrected and smoothed covariance matrices</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>Recon_measurements</td>
<td>Structure to store reconstructed outputs</td>
<td>struct</td>
<td>1x1</td>
</tr>
<tr>
<td>Smooth_measurements</td>
<td>Structure to store smoothed outputs</td>
<td>struct</td>
<td>1x1</td>
</tr>
</tbody>
</table>

Table 4-8: Properties of class EKF with protected access

Methods of the class EKF

The constructor of this class is created as,

\[
EKF_{-}Obj = EKF \left( PROBLEM_{-}Obj \right)
\]

The dynamic methods defined for this class are listed in Table 4-9

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial_EKF ( EKF_Obj)</td>
<td>This function takes EKF_Obj as an input argument and initialize all its properties.</td>
</tr>
<tr>
<td>ARGS = Initial_compile_EKF (EKF_Obj, N)</td>
<td>This method takes EKF_Obj and N as inputs and returns a structure EKF_struct, to be used as entry-point function in compilation. Also input N gives the maximum allowable sampling size.</td>
</tr>
<tr>
<td>Compiler_fun_EKF (EKF_Obj, max_number_samples)</td>
<td>Compiling of external scripts EKF_fun and EKF_Smooth is done by this method, where max_number_samples defines the</td>
</tr>
</tbody>
</table>
maximum sampling size allowed during run time.

Based on the ‘true’ or ‘false’ value of compile_flag, it calls m-file or mex-file to compute reconstructed states and outputs by implementing the EK filtering algorithm. Also, the property smooth lets it decide whether or not to compute smoothed estimates, using extended RTS algorithm.

It extracts and returns the values of reconstructed states, reconstructed covariance and reconstructed outputs from the input argument EKF_Obj.

It returns the smoothed estimates for state, output and covariance from EKF_Obj.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ EKF_Obj ] = EKF_fun_fct_call ( EKF_Obj, compile_flag, smooth)</td>
<td>Based on the ‘true’ or ‘false’ value of compile_flag, it calls m-file or mex-file to compute reconstructed states and outputs by implementing the EK filtering algorithm. Also, the property smooth lets it decide whether or not to compute smoothed estimates, using extended RTS algorithm.</td>
</tr>
<tr>
<td>[ x, P, y ] = gets_function_val ( EKF_Obj)</td>
<td>It extracts and returns the values of reconstructed states, reconstructed covariance and reconstructed outputs from the input argument EKF_Obj.</td>
</tr>
<tr>
<td>[ x_s, P_s, y_s ] = gets_smooth_val ( UKF_Obj)</td>
<td>It returns the smoothed estimates for state, output and covariance from EKF_Obj.</td>
</tr>
</tbody>
</table>

Table 4-9: Methods of class EKF

4.2 Other Auxiliary functions

Beside the functions defined as class methods, other important functions were also created to implement the filtering and smoothing algorithms and to ensure the consistency of function names used in programming. Their purpose of creation is presented below along with a short description.

4.2.1 Function UKF_FUN_AUG_UNSTORED

As mentioned previously, this function is called by class UKF to implement the UK filtering algorithm to obtain the estimated states and reconstructed outputs. One can appreciate the reason behind making it an external function rather than a class method and that is ‘Codegen’ does not support objects (as of yet) and it is easier to compile normal functions than the member functions of a class.

<table>
<thead>
<tr>
<th>Function call</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ UKF_Struct] = UKF_fun_aug_unstored ( UKF_Struct)</td>
<td>This function is based on general application of UKF and hence initializes state vector and covariance matrix in the augmented form. Thereafter, the sigma points are created and propagated thorough nonlinear system to obtain uncorrected estimate for state and covariance. Later on these uncorrected estimates are updated using measurements to obtain the corrected estimates. This whole procedure is repeated till the end of simulation time and finally UKF_Struct is returned with the updated values.</td>
</tr>
</tbody>
</table>

Table 4-10: Function description of UKF_FUN_AUG_UNSTORED
4.2.2 **UKF_SMOOTH**

The purpose to create this function is to obtain smooth estimates for states and outputs based on unscented RTS algorithm. This function is also compiled later on and hence made external on account of the same reason as mentioned for previous function.

<table>
<thead>
<tr>
<th>Function call</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[UKF_Struct] = UKF_Smooth (UKF_Struct)</code></td>
<td>The data for reconstructed states, covariance and cross time covariance ($P_{\hat{x}^f_{k}^f</td>
</tr>
</tbody>
</table>

Table 4-11: Function description of UKF_SMOOTH

4.2.3 **Function EKF_FUN**

Similar to previous function, this function is also called by $EKF$ class to implement the EK filtering algorithm to obtain the estimated states and outputs; and is made external to support ‘Codegen’.

<table>
<thead>
<tr>
<th>Function call</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[EKF_Struct] = EKF_fun (EKF_Struct)</code></td>
<td>It is works on linearization based EKF and first initializes the state vector and covariance matrix as provided by the user. The working procedure is same as mentioned in section 2.3.1. Later on, modified $EKF_{Struct}$ is returned with estimated states, covariance and outputs.</td>
</tr>
</tbody>
</table>

Table 4-12: Function description of EKF_FUN

4.2.4 **EKF_SMOOTH**

This also is an external function and obtains smooth estimates for states and outputs based on extended RTS algorithm.

<table>
<thead>
<tr>
<th>Function call</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[EKF_Struct] = EKF_Smooth (EKF_Struct)</code></td>
<td>The data for estimated states, covariance and state transition matrix ($\Phi_k$) are first extracted and then the modified $EKF_{Struct}$ is returned with smoothed estimates which are evaluated based on the equations (2-76) to (2-80).</td>
</tr>
</tbody>
</table>

Table 4-13: Function description of EKF_SMOOTH

4.2.5 **Function NAME_CHECK**

This function, whenever called, makes sure that the function names as entered by user, to be used for computing system and output equations and also the Jacobians, are same as those used in programming. The function names were used directly in Matlab scripts instead of taking the names as inputs because these Matlab scripts ($m$-files) are later to be compiled and ‘Codegen’ does not support anonymous function names as input arguments.
### Function call

<table>
<thead>
<tr>
<th>Function call</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Names_fun_return = name_check (Names_fun)</td>
<td>This function takes the file names as entered, in the form of a structure, as input. It first compares the entered names with the ones used and if they are consistent it returns the name as it is. Otherwise it retrieves the inconsistent function names, creates new <em>m-files</em> for them with the used function names in a directory called <em>New_functions</em> and subsequently returns the used function names.</td>
</tr>
</tbody>
</table>

*Table 4-14: Function description of NAME_CHECK*
4.3 Re-entry Vehicle Example

The current section illustrates the working of the program using the example problem of the Re-entry vehicle, as already stated in section 2.6. Only the implementation specific description is presented here, and the problem description, along with the obtained results and some remarks can again be seen in section 2.6.

4.3.1 Implementation

Based on the system dynamic equations as mentioned in section 2.6, data for the real states was generated by integrating those system equations over time with $\Delta t = 1/64$ sec. For that, the state vector was initialized as,

$$\mathbf{x}(0) = [3 \times 10^5 \ 2 \times 10^4 \ 10^{-3}]$$ (4-1)

And, as has already been mentioned in the problem description that the body falls freely, hence no inputs were given.

In the actual example as considered in (Simon Julier, 2000), position of the body was measured at discrete points in time using radar capable of measuring range corrupted by Gaussian measurement noise; but for the present application, the measurements were obtained based on the output equation (2-262) with an additive Gaussian measurement noise, generated using 'randn' function in Matlab, which has a covariance $\mathbf{R}$.

Also, for simulation $\gamma$ (constant), $M$ (the Radar height) and $a$ (the horizontal range between the body and radar) were chosen as

$$\gamma = 5 \times 10^{-5}; \ M = 100,000 \ ft; \ a = 100,000 \ ft$$ (4-2)

4.3.2 Set up for application

For filter application, the provided data and given inputs by user are listed in Table 4-15 and Table 4-16 respectively.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$ _perfect</td>
<td>Real states of the system to obtain estimation error.</td>
</tr>
<tr>
<td>$u$</td>
<td>Input data as measured by sensors</td>
</tr>
<tr>
<td>$z$</td>
<td>Measurements obtained over the system</td>
</tr>
<tr>
<td>$t$</td>
<td>Simulation time of the system</td>
</tr>
</tbody>
</table>

Table 4-15: Data provided

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$ _uncorr_0</td>
<td>Initializing state vector</td>
<td>$[3 \times 10^5 \ 2 \times 10^4 \ 3 \times 10^{-5}]^T$</td>
</tr>
<tr>
<td>$P_i$ _uncorr_0</td>
<td>Initializing state covariance matrix</td>
<td>$\begin{bmatrix} 10^6 &amp; 0 &amp; 0 \ 0 &amp; 4 \times 10^6 &amp; 0 \ 0 &amp; 0 &amp; 10 \end{bmatrix}$</td>
</tr>
<tr>
<td>$Q$ _PSD</td>
<td>Process noise power spectral density matrix</td>
<td>$\begin{bmatrix} 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 0 \end{bmatrix}$</td>
</tr>
<tr>
<td>$R$</td>
<td>Measurement noise covariance</td>
<td>$10000 \ ft^2$</td>
</tr>
<tr>
<td>$f$</td>
<td>Function name of system equations</td>
<td>EKF_system_equation</td>
</tr>
<tr>
<td>$h$</td>
<td>Function name of output equations</td>
<td>EKF_output_equation</td>
</tr>
<tr>
<td></td>
<td>Function name to evaluate</td>
<td>EKF_jacobians_&lt;br&gt;name&gt;</td>
</tr>
<tr>
<td>---</td>
<td>------------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>A</td>
<td>system matrix</td>
<td>EKF_jacobians_A</td>
</tr>
<tr>
<td>B</td>
<td>input matrix</td>
<td>EKF_jacobians_B</td>
</tr>
<tr>
<td>F</td>
<td>process noise distribution matrix</td>
<td>EKF_jacobians_F</td>
</tr>
<tr>
<td>C</td>
<td>output matrix</td>
<td>EKF_jacobians_C</td>
</tr>
<tr>
<td>D</td>
<td>feed-through matrix</td>
<td>EKF_jacobians_D</td>
</tr>
<tr>
<td>E</td>
<td>output process noise matrix</td>
<td>EKF_jacobians_E</td>
</tr>
<tr>
<td>G</td>
<td>measurement noise matrix</td>
<td>EKF_jacobians_G</td>
</tr>
</tbody>
</table>

Table 4-16: Inputs by user

**Observation**

A difficulty in tuning the UKF algorithm was observed, which was leading to the failure of the filter after few iterations, if $Q$ and $R$ are not chosen correctly. This is due to the requirement of the covariance matrix ($P_k$) to be positive definite for 'Cholesky decomposition'.
5 Conclusions and perspective

The Flight Path Reconstruction procedure is effectively a nonlinear estimation problem whose preciseness is dependent upon the chosen state estimator to a great extent. Further challenges may arise for the state estimator with the availability of low-quality sensors and for higher levels of process noise during a flight test.

The Extended Kalman Filter has been an extensively used state estimator for the FPR application, but owing to its intrinsic limitation of explicit calculations of Jacobian to linearize the system, the inevitable divergence is merely deferred. On the contrary, the Unscented Kalman Filter is regarded as another approach for a nonlinear stochastic state estimation problem and exhibits certain advantages over EKF due to the propagation of exclusively selected sigma points for approximating the Gaussian distribution rather than truncating the Taylor series to an arbitrary order.

In the application part of this thesis, the EKF and UKF algorithms were first implemented for the FPR of a Re-Entry vehicle example and based on the obtained estimation errors, a performance comparison was made between the two filters in which the UKF reveals better results even for lower sampling rates.

Another execution includes an employment of the two filtering algorithms to a simulation model of an aircraft and equally good estimation performance was achieved by both the filters for the stated design condition with all parameters at their optimum values. To address the posed research questions, the obtained estimation results were investigated with respect to various factors and the observations and possible remedies were illustrated in detail. Beside with the optimum parameters, the two filters show substantially similar results for un-tuned noise matrices, varying measurement frequencies and with changing sampling rates. The superiority in estimation by UKF was only evident in the case of crude approximations for initial states, wherein certain cases divergence was witnessed by EKF but UKF still converges.

For an absolute performance evaluation, explicit application of the algorithms over a real-flight data is necessary. Therefore, for future work it is envisioned to implement the two algorithms on an actual flight test data for additional characterization of the performance benefits of UKF over EKF. There is also a possibility of further extension of the work for real-time flight applications.

END
6 References


A. Appendix A – Fundamental Derivations

Consider a linear, continuous, time-invariant system of the form,

\[ \dot{x}(t) = Ax(t) + Bu(t) + Fw(t) \]
\[ z(t) = Cx(t) + Gv(t) \]  \hspace{1cm} (A-1)

With assumption that process noise vector \( w(t) \) and measurement noise vector \( v(t) \) are zero-mean, white, Gaussian noise. That is,

\[ E[w(t)] = 0; \quad E[w(t)w^T(t)] = I \delta(t - \tau) \]
\[ E[v(t)] = 0; \quad E[v(t)v^T(t)] = I \delta(t - \tau) \]  \hspace{1cm} (A-2)

Since the system considered is stochastic on account of presence of the process noise \( w(t) \) which cannot be measured directly, use of an estimator is sought which is unbiased and efficient enough to reconstruct the real system states and outputs from noisy measurements.

The system considered is hybrid in a way as the dynamic system itself is continuous whereas the measurements from various sensors are discrete. Therefore, the transformation of discrete equations from continuous to discrete time are described in next section.

A.1 Relation between Continuous and Discrete Time Linear Dynamic Systems

The results here are presented in discrete time with sampling instants \( t_k = k \Delta t \). Initializing from time instant \( t_k \), the general solution to the differential equation described in (A-1) is,

\[ x(t) = e^{A(t-t_k)} x(t_k) + \int_{t_k}^{t} e^{A(t-\tau)} (Bu(\tau) + Fw(\tau)) d\tau \]  \hspace{1cm} (A-3)

The solution at time \( t_{k+1} \) is evaluated. Since \( \Delta t \) is small, \( u(t) \) can be assumed constant in the interval \([t_k, t_k + \Delta t]\) and can be taken out of the integral. Also, \( w(t) \) with its discrete counterpart \( \frac{1}{\sqrt{\Delta t}} w_k \) (see reference [1] for more details.) can also be regarded as constant within interval \([t_k, t_k + \Delta t]\) and thus can be taken out as well. This leads to,

\[ x(t_{k+1}) = x_{k+1} \]
\[ = e^{A(t_{k+1}-t_k)} x_k + \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau)} d\tau \cdot Bu_k + \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau)} d\tau \cdot \frac{1}{\sqrt{\Delta t}} Fw_k \]  \hspace{1cm} (A-4)

Taking expressions separately, also substituting \( \Phi(\Delta t) \) for state transition matrix and \( \Psi(\Delta t) \) for discrete input matrix,
\[
\Phi(\Delta t) = e^{\Delta(t_{k+1} - t_k)}x_k = e^{A\Delta t} = \sum_{l=0}^{\infty} \frac{(A\Delta t)^l}{l!}
\]

\[
\Psi(\Delta t) = \int_{t_k}^{t_{k+1}} e^{A(t_{k+1} - \tau)}d\tau = \int_0^{\Delta t} e^{A\nu}d\nu = \sum_{l=1}^{\infty} \frac{A^{l-1}\Delta t^l}{l!}
\]

With \( \nu = t_{k+1} - \tau \) and \( d\nu = -d\tau \)

Thus the discrete equivalent to the continuous time system (A-1) is,

\[
x_{k+1} = \Phi(\Delta t)x_k + \Psi(\Delta t) \cdot Bu_k + \Psi(\Delta t) \cdot F_dw_k
\]

\[
z_k = Cx_k + \frac{1}{\sqrt{\Delta t}}Gv_k = Cx_k + G_dv_k
\]

Where the discrete form of noise processes are defined as,

\[
E[w_k] = 0; \quad E[w_kw_k^T] = I\delta_{kl}
\]

\[
E[v_k] = 0; \quad E[v_kv_k^T] = I\delta_{kl}
\]

### A.2 The Linear Kalman Filter

Based on the principle of a two-step procedure; firstly, it propagates the deterministic part of the states for a given stochastic system and later these erroneous predictions are corrected with the available measurements at regular intervals. According to the nomenclature used here, in representing the state vector \( \tilde{x}_k \) and \( \hat{x}_k \), the superscripts ‘−’ and ‘+’ indicate an ‘uncorrected estimate’ and a ‘corrected estimate’ respectively, at the time instant \( k \). Therefore, using equation (A-7) the state vector is propagated as (leaving out \( \Delta t \) for conciseness)

\[
\tilde{x}_{k+1}^- = \Phi\tilde{x}_k^- + \Psi Bu_k
\]

The obtained estimation errors at two different time instants are

\[
\hat{\epsilon}_k^+ = \tilde{x}_k^+ - x_k
\]

\[
\hat{\epsilon}_{k+1}^- = \tilde{x}_{k+1}^- - x_{k+1}
\]

Where \( x_k \) represents the true state vector at time \( k \). Also the expected value of the above error expression can be shown as

\[
E[\hat{\epsilon}_{k+1}^-] = E[\tilde{x}_{k+1}^- - x_{k+1}]
\]

Taking the expression for the state estimate and the true state from equation (A-9) and (A-7) respectively, the above expected value reduces to

\[
E[\hat{\epsilon}_{k+1}^-] = E[(\Phi\tilde{x}_k^+ + \Psi Bu_k) - (\Phi x_k + \Psi Bu_k + \Psi F_dw_k)] =
\]

\[
= E[\Phi(\tilde{x}_k^+ - x_k) - \Psi F_d w_k] = \Phi E[\hat{\epsilon}_k^+] - \Psi F_dE[w_k]
\]
Appendix A – Fundamental Derivations

One can note here that, the above error expectation $E[\hat{e}_{k+1}^+]$ will converge to zero for stable systems, as $w_k$ is assumed to be zero mean.

The state error covariance matrix is propagated next, in a similar way.

$$P_{k+1}^- = E[\hat{e}_{k+1}^- (\hat{e}_{k+1}^-)^T] = E[(\hat{x}_{k+1}^- - x_{k+1}) \cdot (\hat{x}_{k+1}^- - x_{k+1})^T] \quad (A-14)$$

As done in equation (A-13), the state estimate and the true state are replaced with their respective expressions, here as well and common terms are cancelled, leading to

$$P_{k+1}^- = E[(\Phi (\hat{x}_k^+ - x_k) - \Psi F_d w_k) \cdot (\Phi (\hat{x}_k^+ - x_k) - \Psi F_d w_k)^T] =$$

$$E[(\Phi \hat{e}_k^+ - \Psi F_d w_k) \cdot (\Phi \hat{e}_k^+ - \Psi F_d w_k)^T] \quad (A-15)$$

By simplifying the expressions, using the linearity of the expectations, one can obtain

$$P_{k+1}^- = E[\Phi \hat{e}_k^+(\hat{e}_k^+)^T \Phi^T] - E[\Phi \hat{e}_k^+ w_k \Phi^T] =$$

$$E[\Psi F_d w_k w_k^T \Phi^T] \quad (A-16)$$

Assuming that the state estimation error $\hat{e}_k^+$ and the process noise $w_k$ are uncorrelated, that is $E[\hat{e}_k^+ w_k^T] = E[w_k (\hat{e}_k^+)^T] = 0$ and also keeping in mind that $Cov[w_k] = I$, the following expression for the propagation of the error covariance matrix can be obtained.

$$P_{k+1}^- = E[\Phi \hat{e}_k^+(\hat{e}_k^+)^T \Phi^T] + E[\Psi F_d w_k w_k^T \Phi^T] =$$

$$= \Phi E[\hat{e}_k^+(\hat{e}_k^+)^T] \Phi^T + \Psi F_d E[w_k w_k^T] \Phi^T =$$

$$= \Phi \Phi^T + \Psi F_d F_d^T \Phi^T = \Phi P_k^+ \Phi^T + \Psi F_d F_d^T \Phi^T \quad (A-17)$$

As mentioned in the starting, the erroneous propagated states $\hat{x}_{k+1}^-$ are updated at regular intervals, using the measurements $z_{k+1}$, to obtain the corrected estimates $\hat{x}_{k+1}^+$. This is done by the following idea,

$$\hat{x}_{k+1}^+ = K^+ \hat{x}_{k+1}^- + K z_{k+1} \quad (A-18)$$

In a similar manner as before, the estimation error in the corrected states is obtained as,

$$\hat{e}_{k+1}^+ = \hat{x}_{k+1}^+ - x_{k+1} \quad (A-19)$$

Now, subtracting $x_{k+1}$ on both sides of equation (A-18) and using the results of equation (A-7), another expression for the above equation can be written as,

$$\hat{x}_{k+1}^+ - x_{k+1} = K^+ \hat{x}_{k+1}^- - x_{k+1} + K z_{k+1}$$

$$\hat{e}_{k+1}^+ = K^+ (\hat{x}_{k+1}^- - x_{k+1}) + K(x_{k+1} - x_{k+1}) + K(C x_{k+1} + G_d v_{k+1})$$

$$= K^+ \hat{e}_{k+1}^- + (K^+ K C - I) x_{k+1} + KG_d v_{k+1} \quad (A-20)$$

Taking expectation over both sides,

$$E[\hat{e}_{k+1}^+] = K^+ E[\hat{e}_{k+1}^-] + (K^+ K C - I) E[x_{k+1}] + KG_d E[v_{k+1}] \quad (A-21)$$

To obtain unbiased estimate by the state estimator, $E[\hat{e}_{k+1}^+] = 0$. The first term on the right side approaches zero as already said in (A-13) and the third term is zero based on the
assumption that \(v_{k+1}\) is zero mean. Therefore, to achieve an unbiased estimator the following result should hold.

\[
K^+ = (I - KC)
\]  
(A-22)

This results in the linear combination,

\[
\hat{x}_{k+1} = \left( I - KC \right) \hat{x}_{k+1} + Kz_{k+1} = \hat{x}_{k+1} + K(z_{k+1} - C\hat{x}_{k+1}).
\]

(A-23)

The matrix \(K\) is known as the Kalman Gain matrix and its known value, in the above expression, would lead to the calculation of the corrected estimates \(\hat{x}_{k+1}\). It is a kind of weighting matrix that weighs the influence of the predicted estimate \(\hat{x}_{k+1}\) and measurements \(z_{k+1}\) in obtaining the corrected estimates.

Now, using equation (A-20) with the result of equation (A-22), an expression for the corrected error covariance matrix can be obtained. That is,

\[
P_{k+1}^+ = E[\hat{e}_{k+1}^+ (\hat{e}_{k+1}^+)^T]
\]

\[
= E[\left( (I - KC)\hat{e}_{k+1}^- + KG_d v_{k+1}\right) \left( (I - KC)\hat{e}_{k+1}^- + KG_d v_{k+1}\right)^T]
\]

\[
= (I - KC)E[\hat{e}_{k+1}^- (\hat{e}_{k+1}^-)^T] (I - KC)^T + (I - KC)E[\hat{e}_{k+1}^- v_{k+1}^T] G_d^T K^T
\]

\[
+ KG_d E[v_{k+1}^T (\hat{e}_{k+1}^-)^T] (I - KC)^T + KG_d E[v_{k+1}^T v_{k+1}] G_d^T K^T
\]

\[
= (I - KC)P_{k+1}^- (I - KC)^T + KG_d G_d^T K^T
\]

(A-24)

The idea for obtaining \(K\) is the requirement for obtaining minimum variance estimates. For this, the variance of the estimation error \(\hat{e}_{k+1}^+\), which is represented by the trace of the error covariance matrix \(P_{k+1}^+\), shall be minimised, where,

\[
tr(P_{k+1}^+)=\sum_{i=1}^{n} E[\left( \hat{e}_{k+1}^+ \right)^2]
\]

\[
(A-26)
\]

And \(E[\left( \hat{e}_{k+1}^+ \right)^2]\) represents the expected value of the square of the estimation error, at time \(k + 1\), in the \(i^{th}\) state.

Therefore, by taking the partial derivative of \(tr(P_{k+1}^+)\) with respect to \(K\), an optimal value for the Kalman Gain matrix can be obtained. To do this, the following result from the matrix calculus is used.

\[
\frac{\partial}{\partial A} (tr(ABA^T)) = 2AB
\]

(A-27)
Appendix A – Fundamental Derivations

That is,
\[
\frac{\partial}{\partial K} \text{tr}(P_k^+) = 2 \cdot (I - KC)P_k^+(-C^T) + 2 \cdot KG_d G_d^T K^T = 0 \quad (A-28)
\]

\[
K_{k+1} = P_k^+ C^T (C P_k^+ C^T + G_d G_d^T)^{-1} \quad (A-29)
\]

After some matrix manipulations using equation (A-25) and (A-29), a simpler expression for the corrected error covariance matrix \( P_{k+1}^+ \) can be obtained.

\[
P_{k+1}^+ = (I - KC) P_{k+1}^- \quad (A-30)
\]

The obtained form is easier to implement, but not frequently used for numerical solutions as the resulting update might make the covariance matrix non-symmetric because of round-off errors and finite machine precision.

### A.3 Solving the System Equations

The following continuous time dynamic system with discrete measurements is considered.

\[
\begin{align*}
x'(t) & = f(t, x(t), u(t)); \quad x(t = 0) = x_0 \\
y(t) & = g(t, x(t), u(t)) \\
z_k & = y_k + G_d v_k
\end{align*}
\]

The deterministic system of differential equations can be integrated in time to get the output \( y(t) \), if an initial value for the states \( x_0 \) was known. The simplest method to integrate an ordinary differential equation is the Euler’s method. The basis for this method is that, in a small time interval \( \Delta t \) the function can be assumed to be linear. That’s is

\[
x(t + \Delta t) = x(t) + x(t) \Delta t + O(\Delta t^2) \approx 0 \quad (A-32)
\]

This practically is a very crude approximation and is seldom used, as for mere satisfactory results, very small step size \( \Delta t \) has to be used. Also, this method has a Local truncation error (LTE) that is, the error repeated at each iteration is

\[
LTE \sim O(\Delta t^2) \quad (A-33)
\]

And the Global Truncation Error (GTE), which is the sum of all the LTE’s, is

\[
GTE = \sum_i LTE_i \sim \frac{t - t_0}{\Delta t} O(\Delta t^2) \sim O(\Delta t) \quad (A-34)
\]

Since this method is not used, would not be further elaborated but detailed derivation and implementation can be seen in (Simon, 2006).

To overcome the shortcomings of the above method, multi-step methods have been developed and most famous amongst them would be Runge-Kutta’s methods. The basis for the multi-step methods is to evaluate equation (A-31) not once but multiple times and the resulting update \( x_{k+1} \) will then be a weighted update of the intermediate results.
In the Fourth Order Runge Kutta method, \( f(\cdot) \) is evaluated 4 times, both at different times and state-space. The first evaluation is at the beginning of the given interval \([t_k, t_{k+1}]\)

\[
k_1 = f(t_k, x(t_k), u(t_k)) \tag{A-35}
\]

The second and third evaluation are done in the middle of the interval at \( t_k + \frac{\Delta t}{2} \). One of them is evaluated at a point in state-space which is computed using the previous evaluation \( k_1 \)

\[
k_2 = f(t_k + \frac{\Delta t}{2}, x(t_k + \frac{\Delta t}{2} * k_1), u(t_k + \frac{\Delta t}{2})) \tag{A-36}
\]

And other one is evaluated using the computed state-space by \( k_2 \)

\[
k_3 = f(t_k + \frac{\Delta t}{2}, x(t_k + \frac{\Delta t}{2} * k_2), u(t_k + \frac{\Delta t}{2})) \tag{A-37}
\]

The last evaluation is done at the end of the interval \([t_k, t_{k+1}]\) and using the point in the state-space computed with \( k_3 \)

\[
k_4 = f(t_k + \Delta t, x(t_k + \Delta t * k_3), u(t_k + \Delta t)) \tag{A-38}
\]

As previously mentioned, the resulting update is computed from a weighted sum of the four evaluations with heavier weightage of \( k_2 \) and \( k_3 \)

\[
x_{k+1} = x_k + \frac{\Delta t}{6} (k_1 + 2k_2 + 2k_3 + k_4) \tag{A-39}
\]

It can be shown that the GTE of the above method is of the order \( O(\Delta t^4) \) which give it, its name ‘fourth-order’ method. Even though higher order methods are available, good results have been achieved with this method and is usually used for flight path reconstruction as higher order methods are based on even more functional evaluations which in turn increases the computational burden.

### A.4 Probability density function (pdf) of a random variable (RV)

This section includes a proof to show that, all odd order moments of a zero-mean R.V. with a symmetric probability density function (pdf) are zero.

Consider \( x \) to be a zero-mean R.V. with \( f_X(x) \) as its pdf. Also, symmetric pdf means:

\[
f_X(x) = f_X(-x) \tag{A-40}
\]

Therefore, \( i^{th} \) order moment of \( x \) can be calculated as

\[
E(X^i) = \int_{-\infty}^{\infty} x^i f_X(x) \, dx \tag{A-41}
\]

\[
= \int_{-\infty}^{0} x^i f_X(x) \, dx + \int_{0}^{\infty} x^i f_X(x) \, dx \tag{A-42}
\]
Appendix A – Fundamental Derivations

Now, if \( i \) is odd then \( x^i = -(x)^i \) and also using Equation (A-40),

\[
\int_{-\infty}^{0} x^i f_X(x) dx = - \int_{0}^{\infty} (-x)^i f_X(-x) dx
\]
\[
= - \int_{0}^{\infty} x^i f_X(x) dx
\]

Putting back in (A-42)

\[
E(X^i) = 0
\]

A.5 Equivalence of the Kalman Gain Matrix Expression in EKF and UKF

In the current section, equivalence of the Kalman Gain matrix expression in EKF and UKF is justified regarding the specialized version of UKF as presented in section 2.4.1.2, for a system without feed-through.

Using equation (2-61), the Kalman Gain matrix is evaluated in EKF as (leaving the indices)

\[
K = PC^T(CC^T + R)^{-1}
\]

(A-45)

And in UKF, its expression is,

\[
K = P_{xy}(P_{yy})^{-1}
\]

(A-46)

The above expression can also be written as,

\[
K = Cov[x, z](Cov[z, z])^{-1}
\]

(A-47)

On individual expansion of each term and using the relation of equation (2-32) ignoring the feed-through terms,

\[
Cov[x, z] = Cov[x, Cx + \nu] = E[(Cx + \nu)^T]
\]

(A-48)

Also, it is assumed that the process is stationary i.e. \( E[x \cdot \nu^T] = 0 \). Therefore, the above expression reduces to,

\[
Cov[x, z] = E[x(Cx)^T] = E[x(x)^T]C^T
\]
\[
= Cov[x]C^T
\]

(A-49)

\[
= PC^T
\]

Similarly, the second expression of equation (A-47) can be written as,

\[
Cov[z, z] = E[(Cx + \nu)(Cx + \nu)^T]
\]
\[
= E[Cx(Cx)^T] + E[\nu(\nu)^T] + E[\nu \cdot (Cx)^T] + E[\nu \cdot \nu^T]
\]

(A-50)
On applying the previously stated assumption,

\[
\text{Cov}[z, z] = E[Cx(Cx)^T] + E[v \cdot v^T] \\
= CE[x(x)^T]C^T + E[v \cdot v^T] \\
= CPC^T + R
\]  

(A-51)

Therefore, by substituting equation (A-51) and (A-49) back in equation (A-47) the final expression of Kalman Gain matrix, as used in UKF, can be obtained.

\[
K = PC^T(CPC^T + R)^{-1}
\]  

(A-52)

Hence on comparing equation (A-52) with (A-45), the required justification is achieved.
B. Appendix B – Additional Figures

B.1 Smoother Results

The following figures are for the smoothed estimates using Extended RTS and Unscented RTS.

Figure B-1: Smoothed States – Position

Figure B-2: Absolute Estimation Error - Position
Appendix B – Additional Figures

Figure B-3: Smoothed States – Kinematic velocities

Figure B-4: Absolute Estimation Error – Kinematic velocities
Figure B-5: Smoothed States – Attitude angles (Euler angles)

Figure B-6: Absolute Estimation Error – Attitude angles
Appendix B – Additional Figures

Comparison of EKF and UKF for Flight Path Reconstruction in System Identification

Mohit Mehndiratta

Figure B-7: Smoothed States – Wind velocities

Figure B-8: Absolute Estimation Error – Wind velocities
Appendix B – Additional Figures

Comparison of EKF and UKF for Flight Path Reconstruction in System Identification

Figure B-9: Smoothed Outputs – Position

Figure B-10: Smoothed Outputs – Kinematic velocities
Figure B-11: Smoothed Outputs – Air data sensor quantities
B.2 Poor Initial Guess

Estimated states and the reconstructed outputs are presented, which are obtained for the poor estimates of the initial states, with initial error covariance matrix to be a diagonal matrix of value $10^{-2}$ for all the states.

![Figure B-12: States – Position with varying Initial conditions](image)

![Figure B-13: States – Kinematic Velocities with varying Initial conditions](image)
Figure B-14: States – Attitude angles with varying Initial conditions

Figure B-15: States – Wind Velocities with varying Initial conditions
Comparison of EKF and UKF for Flight Path Reconstruction in System Identification

Mohit Mehndiratta

Appendix B – Additional Figures

Figure B-16: Outputs – Position with varying Initial conditions

Figure B-17: Outputs – Kinematic Velocities with varying Initial conditions
Appendix B – Additional Figures

Figure B-18: Outputs – Air Data quantities with varying Initial conditions

Figure B-19: Average MSE for percentage variation in Initial conditions
B.3 Results For Variation in R

Estimated states and the reconstructed outputs for the variation in $R$ matrix alone are shown.

![Graphs of Position and Velocities with varying $R$](image)

Figure B-20: States – Position with varying $R$

![Graphs of Position and Velocities with varying $R$](image)

Figure B-21: Kinematic Velocities with varying $R$
Comparison of EKF and UKF for Flight Path Reconstruction in System Identification

Appendix B – Additional Figures

Figure B-22: States – Attitude angles with varying $R$

Figure B-23: States – Wind Velocities with varying $R$
Appendix B – Additional Figures

Comparison of EKF and UKF for Flight Path Reconstruction in System Identification

Mohit Mehndiratta

Figure B-24: Outputs – Position with varying $R$

Figure B-25: Outputs – Kinematic Velocities with varying $R$
Figure B-26: Outputs – Air Data quantities with varying $R$