COMPARISON OF EXTENDED KALMAN FILTER AND UNSCENTED KALMAN FILTER FOR CONTROL MOMENT GYROSCOPE(CMG) INVERTED PENDULUM

by

Jyot R. Buch

A project report submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Electrical Engineering

Charlotte

2016

Approved by:

____

Dr. Yogendra Kakad

____

Dr. Robert Cox

____

Dr. Tao Han

ABSTRACT

JYOT R. BUCH. Comparision of EKF and UKF for CMG Inverted Pendulum.
(Under the direction of DR. YOGENDRA KAKAD)

This research project investigates nonlinear state estimation for Control Moment Gyro (CMG) inverted pendulum system. The control moment gyroscope inverted pendulum is originally modeled at Embry-Riddle Aeronautic University by Dr. Douglas Isenberg. Extended Kalman Filter(EKF) and Unscented Kalman Filter(UKF) are very well known techniques derived for nonlinear state estimation from the originally presented kalman filter for LTI systems. Since then, its been considerable research focus among the control and estimation community for investigation regarding nonlinear state estimation, using EKF or UKF. Control Moment Gyro Inverted Pendulum is inherently non-linear system with stiff nonlinearity. In this report, the dynamics for the same are revisited and nonlinear state estimation approach using EKF and UKF is summarized along with their MATLAB implementations.

Here, the estimate is used for designing a controller using state feedback. Out of these algorithm for state estimation, which one will perform better, depends on various aspects of model nonlinearity and most of the time unpredictable to comment on. Results for both algorithms are discussed and compared to fulfill the objective of the project.

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere thanks to my very respected advisor, Dr. Yogendra P. Kakad for throughout support, guidance and encouragement. I am grateful for his trust and giving me an opportunity to work with him. His continuous guidance during my masters course work and research project has been very helpful to me. His guidance for this research project during my internship at Mathworks has been phenomenal.

I would like to express gratitude to my committee members, Dr. Robert William Cox and Dr. Tao Han, for their participation and spending time to assess my work.

I feel sad that Dr. Bharat Joshi, who was also in my committee and provided immense support and guidance during my masters graduate program, he passed away before the project presentation. I truly appreciate his friendly advises as well as his formal role in department as graduate program director. He will be always in memory of his colleagues, friends and family.

I would like to thank Dr. Yawo Amengenu for helping me with most of the my questions regarding to CMG inverted pendulum, since he spent considerable time on it during his Ph.D..

I am very thankful to Dr. Arda Ozdemir (MathWorks Inc.) and Dr. Micheal Saginaw for their guidance regarding MATLAB script implementation of EKF and UKF.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Kalman Filtering and State Estimation Approach

State estimation is very well known approach in control theory. It is a set of mathematical equations which provides an estimate of the internal state based on the model of the system, plant's input information and output measurements [1]. This technique is known as state observer design, in which often the goal is to find out observer gain, which can be used to compute state estimate. By using that estimate, one can design a controller using state feedback and apply other advanced control strategies.

Over the years quest has been to produce estimate which is *"optimal"* in sense of estimation error i.e. it minimizes estimation error. Kalman filtering is one of the most discussed algorithm in this area in which the estimate is calculated based on a series of measurements containing statistical noise and other model uncertainties (plant noise) observed over time. It tries to estimates unknown variables that tend to be more precise than those based on a single measurement. This revolutionary theory was named after one of the primary developer Rudolf E. Kalman [2][3]. Since, its introduction in 1960, it has been applied and investigated in various areas such as aerospace, automobile, finance, engineering, control, robotics, biological studies, medicine, image processing etc. along with various versions which were derived later.

Kalman filter was proposed as new approach to linear filtering and prediction problem in [4]. It is optimal in a sense that it minimizes the mean value of the sum of estimation errors and error covariance. Generally, most of the systems in real life are inherently non-linear, and for them derivatives of kalman filter works efficiently. These derivatives tend to either linearize the nonlinear model around estimated points i.e. Extended Kalman Filter [5] or use nonlinear transformation i.e. Unscented Kalman Filter [6]. In general, kalman filter is good choice not only in theory but also in hardware implementation as it gives very negligible error in performance along with computationally efficient optimal solution. This research project investigates both the algorithms for control moment gyro inverted pendulum. MATLAB scripts are written to generate the simulation results to compare EKF and UKF.

1.2 Assumptions

There are certain assumption made for this work which are mentioned bellow.

Assumption 1:

This work has been done considering Gaussian assumption of noise and model uncertainties. White noise present in almost all measurement, Gaussian noise assumption works for most of the cases, but it is not always correct. Model Uncertainties are mostly due to modeling errors, model parameter change due to environment or process variables.

Discrete-time version of Kalman Filter is more used for practical implementations, but the scope of the project is to get simulation results, so continuous time version will be used.

Assumption 2:

Generally, Noise can interfere in various ways to the plant dynamics and measurement, but in this work the noise is considered to be *"Additive"* in nature.

Assumption 3:

A necessary condition for the kalman filter to work correctly is that the system for which states are to be estimated, is observable[3]. Therefore, it is the first step in design to check the observability of the system. For specific small linearizion step, nonlinear system behave as linear dynamical system, hence linear dynamics are discussed bellow.

The continuous time linear state space system represented as,

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{1.1}$$

is observable only if it is possible to determine the initial state of the system given the knowledge of input and output.

$$M_{obs} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{1.2}$$

This is possible only if observability matrix $\boldsymbol{M_{obs}}$ is full rank matrix with rank n (the number of state variables).

Similarly, complete state controllability describes the ability of an external input (the vector of control variables) to move the internal state of a system from any initial state to any other final state in a finite time interval. **To design a controller** it is necessary to check the controllability of the system,

$$M_{ctr} = \begin{bmatrix} B & AB & A^2B & & A^{n-1}B \end{bmatrix} \tag{1.3}$$

System **is controllable** only if controllability matrix $\boldsymbol{M_{ctr}}$ is full rank with rank n. For, *Nonlinear System* these definitions are explained in more detail [7] by yawo, which are out of the scope of this project. Since, Observability is the main concern for Kalman filter design, following are several pit-falls due to *non-observability* of the system,

- The Transfer function from the input variable y to the output variable y has an order that is less then the number of state variables (n).

- There are state variables or linear combinations of state variables that do not show any response
- The Steady-state value of the Kalman Filter gain can not be computed. This gain is used to update the state estimates from measurement of the system.

In this work, it is assumed that Control Moment Gyro Inverted Pendulum system is fully controllable and observable.

Assumption 4:

There are various derived Kalman filtering algorithms both in discrete time and continuous time, in this work continuous-time Extended Kalman-bucy filter and Unscented Kalman-bucy filter are implemented in MATLAB. Due to inherent programing style in MATLAB, the continuous time has been devided in fix time steps and ODE's are solved in those time durations, so that continuous time model dynamics can be used for estimation, i.e. no need for discretization.

1.3 Organization of Report

This report contains 5 sections, which are summarized bellow.

Chapter 1: deals with introduction, and assumption made for this research study.

Chapter 2: gives the detailed explanation on Extended Kalman Filter for Nonlinear system.

Chapter 3: enlists the limitation of Extended Kalman Filter and discusses Unscented Kalman Filter with necessary equations

Chapter 4: summarizes the Control Moment Gyro Inverted Pendulum Dynamics.

Chapter 5: discusses the results and MATLAB implementation of these algorithms.

CHAPTER 2: EXTENDED KALMAN FILTERING

Extended Kalman filter is the extension of firstly introduced linear system Kalman Filter [3]. Among various versions, continuous time EKF is also known as Extended Kalman-Bucy filter [8]. Fundamentally, Extended Kalman Filter linearizes the state dynamics around the current estimate using first order taylor series expansion, which is explained in 2.1 and 2.2.

2.1 Introduction

Considering the Nonlinear state space system,

$$\begin{aligned} \dot{x} &= f(x, u, w, t) \\ y &= h(x, v, t) \end{aligned} \tag{2.1}$$

Where, f is nonlinear state transition function, which involves states and Gaussian noise/uncertainty for state as well as measurement.

$$\dot{x} = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \vdots \\ \dot{x_n} \end{bmatrix} = \begin{bmatrix} f_1() \\ f_2() \\ \vdots \\ f_n() \end{bmatrix} \tag{2.2}$$

y = *h()* is the measurement vector function with r measurements

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_r \end{bmatrix} = \begin{bmatrix} h_1() \\ h_2() \\ \vdots \\ h_r() \end{bmatrix} \tag{2.3}$$

- $\boldsymbol{w(t) \sim N(0,Q)}$ represents model uncertainty or plant/process noise vector with 0 mean and Q auto-covariance, *i.e.* $E[w] = 0, Q = E[ww^T]$

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \tag{2.4}$$

  The standard assumption is that,

$$Q = \begin{bmatrix} Q_{11} & 0 & 0 & 0 \\ 0 & Q_{22} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & Q_{nn} \end{bmatrix} = diag(Q_{11}, Q_{22}, ..., Q_{nn}) \tag{2.5}$$

  Where, Q_{ii} is variance of w_i.

- $\boldsymbol{v(t) \sim N(0,R)}$ is (random) measurement noise vector with 0 mean and R auto-covariance, *i.e.* $E[v] = 0, R = E[vv^T]$

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \tag{2.6}$$

The standard assumption is that,

$$R = \begin{bmatrix} R_{11} & 0 & 0 & 0 \\ 0 & R_{22} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & R_{rr} \end{bmatrix} = diag(R_{11}, R_{22}, ..., R_{rr}) \tag{2.7}$$

Where, R_{ii} is variance of v_i

The "Strength" or power of these noise can be adjusted by Q and R variances.

- Linearizion around current estimate is performed by *Tailor series* expansion:

$$f(x, u, w, t) = f(\hat{x}, u_0, w_0, t) + \left.\frac{\partial f}{\partial x}\right|_{\hat{x}} (x - \hat{x}) + ... \tag{2.8}$$

Note that, we are considering only first order Taylor series approximation i.e. ignoring higher order terms.

- If $\hat{x}$ is an estimate provided by any estimator, then *estimation error* is given by,

$$e(t) = x - \hat{x} \tag{2.9}$$

Error (auto)covariance is given by,

$$P(t) = E[e(t)e(t)^T] = E[(x - \hat{x})(x - \hat{x})^T] \tag{2.10}$$

2.2 Extended Kalman-Bucy Filter Algorithm

1. This is the *initial step* and the operations are executed only once. Assuming the initial guess of the state is x_0

$$\hat{x}(0) = \hat{x}_0 = x_0 = E(x(0)) \tag{2.11}$$

2. Co-variance of Estimation Error is P with initial value to be $P0$ (*initial step*)

$$P_0 = P(0) = E[(x(0) - \hat{x}(0))(x(0) - \hat{x}(0))^T] \tag{2.12}$$

3. Following Matrices are computed

$$\tilde{Q} = LQL^T \tag{2.13}$$

$$\tilde{R} = MRM^T \tag{2.14}$$

4. Following partial derivatives matrices are evaluated at current estimate:

$$A = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}} \tag{2.15}$$

$$L = \left.\frac{\partial f}{\partial w}\right|_{\hat{x}} \tag{2.16}$$

$$C = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}} \tag{2.17}$$

$$M = \left.\frac{\partial h}{\partial v}\right|_{\hat{x}} \tag{2.18}$$

5. Predict Step:

Algebraic Riccati Equation for error covariances,

$$\dot{P} = AP + PA^T + \tilde{Q} - PC^T \tilde{R}^{-1} CP \tag{2.19}$$

Prediction Step : Propagate the state ahead,

$$\dot{x} = f(\hat{x}, u, 0) \tag{2.20}$$

6. Correct Step:

   Kalman Gain is computed by following equation

$$G = PC^T \tilde{R}^{-1} \tag{2.21}$$

Calculate the estimated measurement,

$$\hat{y} = h(\hat{x}, v_0, t) \tag{2.22}$$

Feedback-Correction step : Correct the state estimate,

$$\dot{\hat{x}} = f(\hat{x}, u, w_0, t) + G \underbrace{(y - \hat{y})}_{\text{Innovation variable/Estimation Error}} \tag{2.23}$$

Step 4, 5 and 6 are repeated iteratively for each linearizion step.

Generally, state auto-covariances and Kalman filter gain converges to steady state value and it is very common practice to use steady state values of the kalman gain. For nonlinear systems, linearized input matrix A will vary with the operating point, and kalman gain G is re-calculated every linearizion time step. As explained above, we designed the optimal estimator or kalman filter of following type,

$$\dot{\hat{x}} = f(\hat{x}, u, w_0, t) + G(y - C\hat{x}) \tag{2.24}$$

Considering, equation 2.1 and 2.9, the error dynamics are given by,

$$\dot{e} = \underbrace{f(x, u, w, t) - f(\hat{x}, u, w_0, t)}_{F(e,\hat{x},u,w,t)} - GC(x - \hat{x}) \tag{2.25}$$

$$\dot{e} = F(e, \hat{x}, u, w) - GCe \tag{2.26}$$

Linearizing around current estimate $\hat{x}$,

$$\hat{e} = \frac{\partial F}{\partial e} e + \underbrace{F(0, \hat{x}, u, 0, t)}_{=0} + \underbrace{\frac{\partial F}{\partial w} w}_{\text{noise}} - GCe + ... \tag{2.27}$$

$$\hat{e} \approx (\tilde{A} - GC)e + \tilde{F}w \tag{2.28}$$

Where,

$$\tilde{A} = \left.\frac{\partial F}{\partial e}\right|_{(0,\hat{x},u,0)} = \left.\frac{\partial f}{\partial x}\right|_{(\hat{x},u,0)} \tag{2.29}$$

$$\tilde{F} = \left.\frac{\partial F}{\partial w}\right|_{(0,\hat{x},u,0)} = \left.\frac{\partial f}{\partial w}\right|_{(\hat{x},u,0)} \tag{2.30}$$

Hence, the dynamics of Kalman Filter can be analyzed by calculating the eigenvalues,

$$[\lambda_1, \lambda_2, ..., \lambda_n] = eig(A - GC) \tag{2.31}$$

It can be shown that Kalman filter is always *asymptotically stable* dynamic system, and all the eigenvalues lies in left half of s-plane.

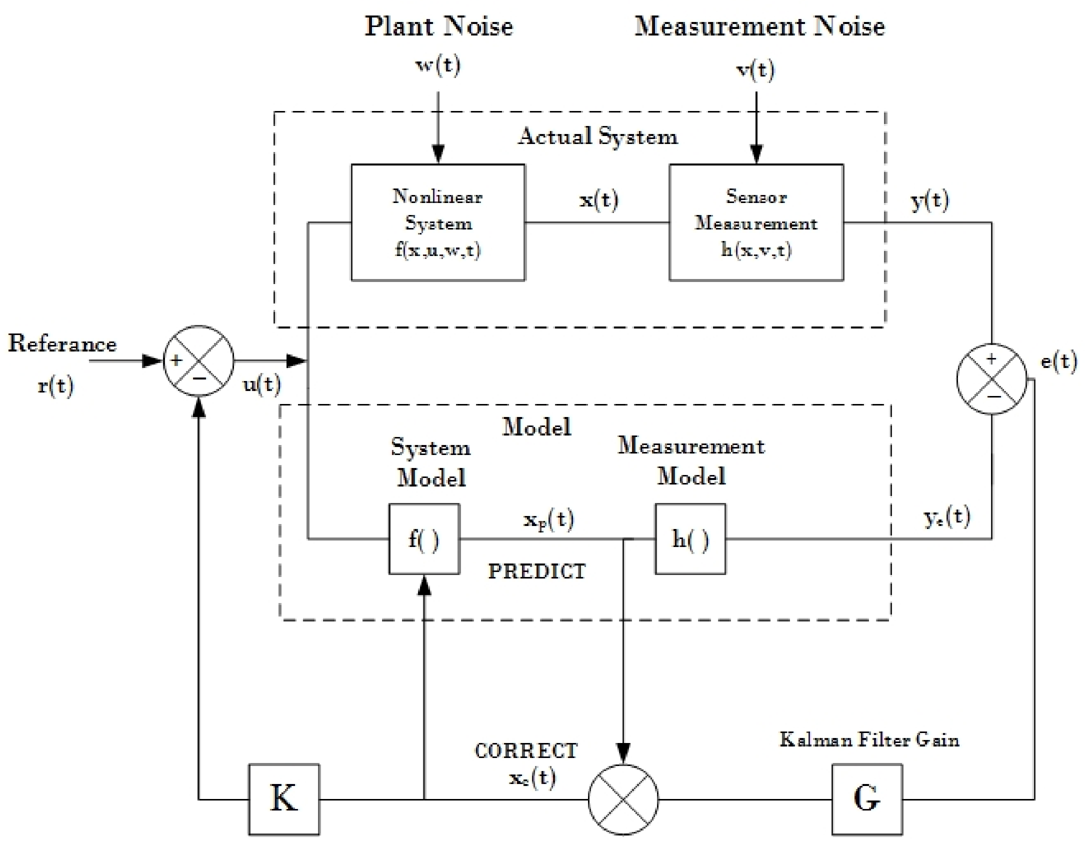

Figure 2.1: Kalman Filter Block Diagram

2.3 Design of Controller

As explained earlier, since system is controllable, we can design full state feedback controller as shown in 2.1 with closed form control law given as,

$$u = -K\hat{x} \tag{2.32}$$

$\hat{x}$ is the estimated state, which is being used here for state feedback. We assume that reference r is kept as 0, hence feedback is directly applied as input. Controller gain K can be calculated using pole placement technique, but care should be taken that the estimate should actually be used by the controller to provide control input, for that it is general practice that controller poles are 4 to 5 times of the observer poles, so that estimator is fast enough.

Gain K can be computed by *place* command in MATLAB,

$$\text{Desired location of controller poles(eigenvalues)} = \frac{eig(A - GC)}{5} \tag{2.33}$$

$$K = place(A - GC, B, eigenvalues) \tag{2.34}$$

Note that one can design LQG (Linear Quadratic Gaussian) control policy for further optimize control law with respect to input and state step cost in linearized dynamics, but we limit our discussion to simple state feedback for control design.

CHAPTER 3: UNSCENTED KALMAN FILTERING

3.1 Limitations of Extended Kalman Filtering

In highly nonlinear problems, the EKF tends to be very inaccurate and underestimates the true covariance of the estimated state. This can lead to poor performance and filter divergence. This is because EKF relies on the linearizion to propagate the mean and covariances of the state. According to [3][9], it can be proved with simple example that generally, UKF gives better results then EKF because of deterministic sampling.

3.2 Unscented Transform

The Unscented transform (UT) is the method for calculating the statistics of a random variable which undergoes nonlinear transformation [11]. Random variable x (dimension L) is propagated by nonlinear function, $y = f(x)$. Where, it is assumed that x has mean $\bar{x}$ and covariance P_x, to calculate the statistics of y, we form a matrix $\mathcal{X}$ which has $2n + 1$ sigma vectors $\mathcal{X}_i$ according to the following:

$$\mathcal{X}_i = \begin{cases} \bar{x}, & i = 0. \\ \bar{x} + \sqrt{(n+\lambda)P_x}, & i = 1,\dots,n \\ \bar{x} - \sqrt{(n+\lambda)P_x}, & i = n+1,\dots,2n \end{cases} \tag{3.1}$$

where,

$$\lambda = \alpha^2(n + \kappa) - n \tag{3.2}$$

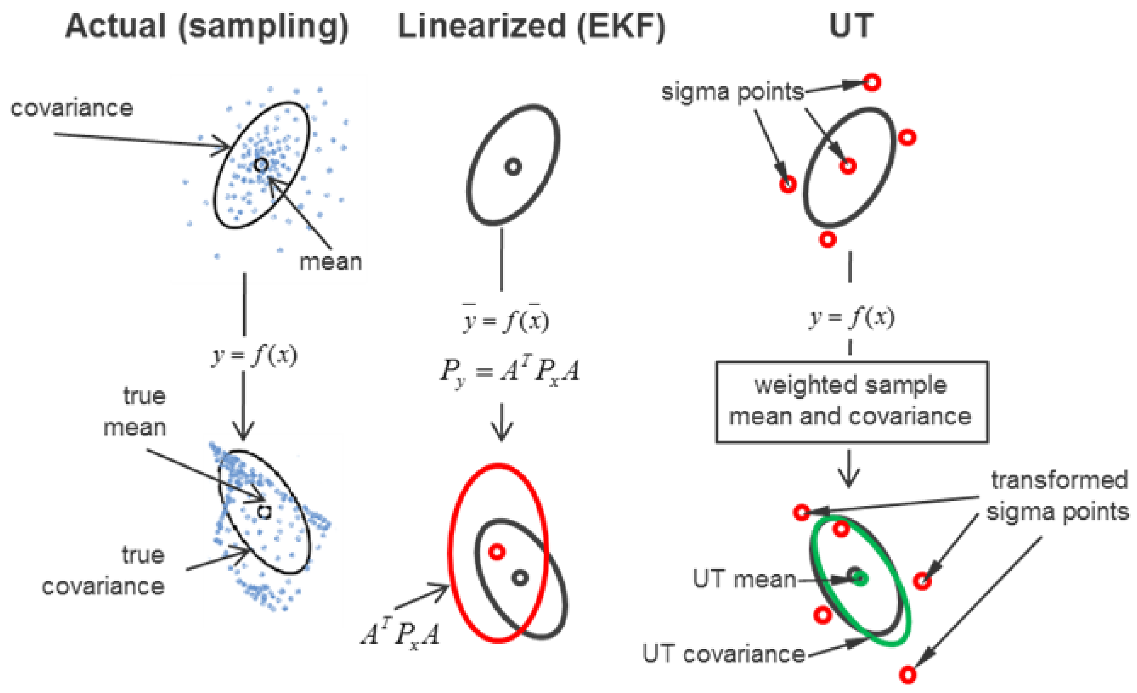

Figure 3.1: Comparison of Filtering Algorithms[10]

The parameters α and κ determines the spread of the sigma points around the mean. $(\sqrt{(n+\lambda)P_x})_i$ is ith column of matrix square root (i.e. Cholesky factorization) sigma points vectors are the column of the sigma point matrix, which are propagated through nonlinear function,

$$\mathcal{Y}_i = f(\mathcal{X}_i) \tag{3.3}$$

The mean and the covariance for y are approximated using a weighted sample mean and covariance of the posterior sigma points, in matrix notation it can be written as,

$$\bar{y} \approx W_m Y \tag{3.4}$$

$$P_y \approx Y W_c Y^T \tag{3.5}$$

Weight matrix is formed as shown bellow,

$$
\begin{aligned}
W_0^{(m)} &= \frac{\lambda}{(n+\lambda)} \\
W_0^{(c)} &= \frac{\lambda}{(n+\lambda)} + (1-\alpha^2+\beta) \\
W_i^{(c)} = W_i^{(m)} &= \frac{1}{2(n+\lambda)}
\end{aligned}
\tag{3.6}
$$

Where, n is number of states. Above method can be used to compute the mean and covariances of states. Full algorithmic steps are mentioned in section 3.3.

3.3 Unscented Kalman-Bucy Filter Algorithm

Consider, the nonlinear state space system,

$$
\begin{aligned}
\dot{x} &= f(x,u,w,t) \\
y &= h(x,v,t)
\end{aligned}
\tag{3.7}
$$

Where, $x(t)$ is the state, $y(t)$ is the measurement, $w(t) \sim N(0,Q)$ is the Gaussian process noise and $v(t) \sim N(0,R)$ is the Gaussian measurement noise. $f()$ is the dynamic model function and $h()$ is the measurement function.

1. This is the *initial step* and the operations are executed only once. Assuming the initial guess of the state is normal distribution with known mean $\bar{x}$ and $P(0) = P_0$ covariance,

$$x_0 \sim N(\bar{x}, P_0) \tag{3.8}$$

2. Following Matrices are computed

$$\tilde{Q} = LQL^T \tag{3.9}$$

$$\tilde{R} = MRM^T \tag{3.10}$$

3. Predict Step :

   For n^{th} order system, sigma point matrix $\mathcal{X}(t)$ is formulated which has *2n+1* elements,(Note: MATLAB indexing starts from 1)

$$\mathcal{X}(t)_i = \begin{cases} \bar{x}(t), & i = 0. \\ \bar{x}(t) + \sqrt{(n+\lambda)P(t)_x}, & i = \mathit{1,2,...,n} \\ \bar{x}(t) - \sqrt{(n+\lambda)P(t)_x}, & i = \mathit{n+1,n+2,...,2n} \end{cases} \tag{3.11}$$

Propagate the sigma points through dynamic model, by solving following differential equation which has solution $\hat{\mathcal{X}}_p$ i.e. predicted sigma points

$$\frac{d\mathcal{X}_i}{dt} = \dot{\mathcal{X}}_i = f(\mathcal{X}(t)_i, u), \quad i = 1, 2...2n+1 \tag{3.12}$$

Compute the predicted mean,

$$\bar{x}_p = \hat{\mathcal{X}}_p w_m = \bar{x}_p(t) \tag{3.13}$$

Compute the predicted covariances,

$$P_p = \hat{\mathcal{X}}_p W \hat{\mathcal{X}}_p^T + \tilde{Q} \tag{3.14}$$

4. Measurement Correct/ Update Step

   Form the predicted values sigma point matrix,

$$\mathcal{X}_p(t)_i = \begin{cases} \bar{x}_p(t), & i = 0. \\ \bar{x}_p(t) + \sqrt{(n+\lambda)P_p(t)_{x_p}}, & i = \mathit{1,2,...,n} \\ \bar{x}_p(t) - \sqrt{(n+\lambda)P_p(t)_{x_p}}, & i = \mathit{n+1,n+2,...,2n} \end{cases} \tag{3.15}$$

   Propagate the sigma points through measurement model

$$\hat{\mathcal{Y}}_i = h(\mathcal{X}_p(t)_i, u), \quad i = 1, 2...2n+1 \tag{3.16}$$

Compute the predicted mean μ,

$$\mu = \hat{\mathcal{Y}} w_m = E(\hat{y}) \tag{3.17}$$

Compute the predicted covariances of the measurement,

$$P_{yy} = \hat{\mathcal{Y}} W \hat{\mathcal{Y}}^T + \tilde{R} \tag{3.18}$$

Compute cross-covariance of the state and measurement,

$$P_{xy} = \mathcal{X}_p W \hat{\mathcal{Y}}^T \tag{3.19}$$

Compute the Kalman Filter Gain G

$$G = \frac{P_{xy}}{P_{yy}} = P_{xy} P_{yy}^{-1} \tag{3.20}$$

Compute filtered state mean and covariance,

$$\bar{x} = \bar{x}_p + G[y - \mu] \tag{3.21}$$

$$P = P_p - G P_{yy} G^T \tag{3.22}$$

Step 3 and 4 are repeated iteratively to compute the estimate at each simulation time step. For the controller design using pole placement technique, here matrices A and B are not available, for that alternative computation method is explained in section 4.3.

CHAPTER 4: CONTROL MOMENT GYRO INVERTED PENDULUM

4.1 Introduction

Control Moment Gyro Inverted Pendulum is shown in the figure 4.1, the pendulum′s pivot point is inertially fixed and actuation is accomplished via the controlled rotation of a massive disk attached to the pendulum about an axis parallel to the pendulums pivot axis. Moreover, the massive rotating disk is allowed to also rotate about an axis that is parallel to the length of the pendulum. Such a mechanism forms a Control Moment Gyroscope (CMG) which can be utilized to provide a torque on the pendulum. The dynamics of the control moment gyroscope derived by Yawo using Hamiltons principle via the Euler-Lagrange equation [12]. Here, this dynamics will be revisited and Nonlinear state estimation approach will be explored.

4.2 Dynamic Model: Control Moment Gyro Inverted Pendulum

As shown in the figure three bodies in this system, forms three degrees-of-freedom associated with the pendulums rotation through angle θ_1, the CMGs rotation through angle θ_2, and the CMG disks rotation through angle θ_3. A vector of generalized coordinates for this system will be thus,

$$\underline{\gamma} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \tag{4.1}$$

The second-order differential equation of the system can be written as,

$$H(\underline{\gamma})\ddot{\underline{\gamma}} + \underline{D}(\underline{\gamma}, \dot{\underline{\gamma}}) + \underline{G}(\underline{\gamma}) = \underline{\tau} \tag{4.2}$$

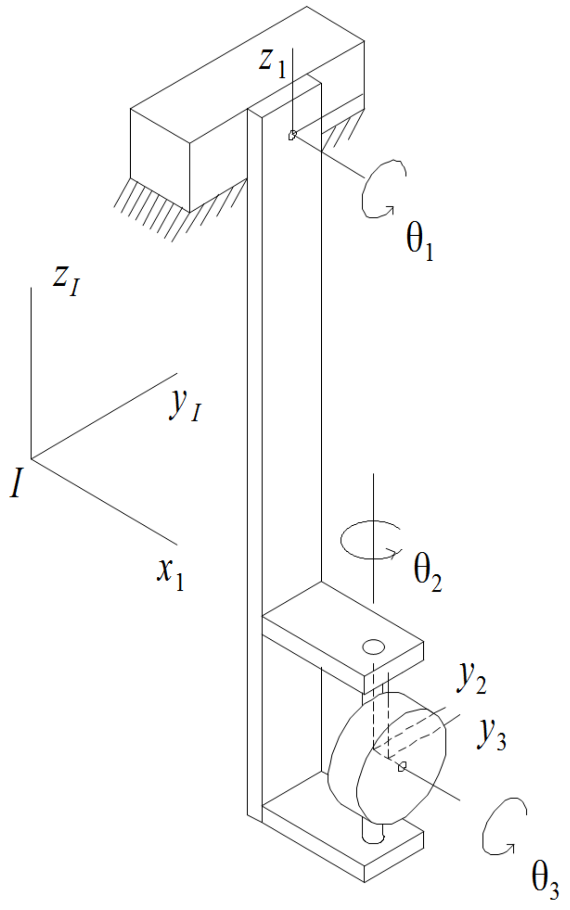

Figure 4.1: Control Moment Inverted Pendulum [12]

Where, symmetric positive-definite system mass matrix H is given by,

$$H(\underline{\gamma}) = \begin{bmatrix} a + b\sin^2(\theta_2) & c\cos(\theta_2) & J_{3xx}\cos(\theta_2) \\ c\cos(\theta_2) & d & 0 \\ J_{3xx}\cos(\theta_2) & 0 & J_{3xx} \end{bmatrix} \tag{4.3}$$

The vector of the generalized Coriolis and Centripetal forces D is given by,

$$\underline{D}(\underline{\gamma}, \dot{\underline{\gamma}}) = \begin{bmatrix} 2b\dot{\theta}_1\dot{\theta}_2\cos(\theta_2)\sin(\theta_2) + J_{3xx}\dot{\theta}_2\dot{\theta}_3\sin(\theta_2) - c\dot{\theta}_2^2\sin(\theta_2) \\ -b\dot{\theta}_1^2\cos(\theta_2)\sin(\theta_2) + J_{3xx}\dot{\theta}_1\dot{\theta}_3\sin(\theta_2) \\ -J_{3xx}\dot{\theta}_1\dot{\theta}_2\sin(\theta_2) \end{bmatrix} \tag{4.4}$$

The vector of generalized gravitational forces G is given by,

$$\underline{G}(\underline{\gamma}) = \begin{bmatrix} Ag\sin(\theta_1) + Bg\cos(\theta_1)\sin(\theta_2) \\ Bg\sin(\theta_1)\cos(\theta_2) \\ 0 \end{bmatrix} \tag{4.5}$$

Where,

$$\begin{aligned} a &= J_{1xx} + J_{2xx} + J_{3xx} + d_1^2 m_3 \\ b &= J_{3yy} - J_{3xx} + d_3^2 m_3 \\ c &= d_1 d_3 m_3 \\ d &= m_3 d_3^2 + J_{2zz} + J_{3yy} \\ A &= d_1 m_2 - \Gamma_{1z} + d_1 m_3 \\ B &= d_3 m_3 \end{aligned} \tag{4.6}$$

In above equation, d_1 is the distance along the z-axis of the frame 1 from the origin of the frame 1 to the origin of the frame 2, d_2 is the distance along the x-axis of the frame 1 from the origin of the frame 1 to the origin of the frame 2, and d_3 is the distance from the origin of the frame d_2 to the origin of frame 3.

Converting in state space variables,

$$\begin{aligned} x_1 &= \theta_1 \\ x_2 &= \dot{\theta}_1 = \omega_1 \\ x_3 &= \theta_2 \\ x_4 &= \dot{\theta}_2 = \omega_2 \\ x_5 &= \theta_3 \\ x_6 &= \dot{\theta}_3 = \omega_3 \end{aligned} \tag{4.7}$$

Hence, nonlinear state space can be written as,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_2 \\ f_2() \\ x_4 \\ f_4() \\ x_6 \\ f_6() \end{bmatrix} = f(x,u) \tag{4.8}$$

Where,

$$\begin{bmatrix} f_2() \\ f_4() \\ f_6() \end{bmatrix} = \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} = \ddot{\underline{\gamma}} = H^{-1}(\underline{\gamma})[\underline{\tau} - \underline{D}(\underline{\gamma}, \dot{\underline{\gamma}}) - \underline{G}(\underline{\gamma})] \tag{4.9}$$

4.3 Implementation

Above dynamics equations [13] are used in formulating the MATLAB script for EKF and UKF, which is attached as appendix. Specific details are mentioned bellow.

- In case of EKF, jacobians are calculated offline using MATLAB Symbolic Math Toolbox and they are supplied in the algorithm, which changes iteratively during the linearizion step.

- CMG inverted pendulum system has one of the eigen value as 0 always, which is refereed to as *Rigid body modes.* So, generally, it gives a natural frequency of zero or close to zero. While designing the control law at each linearizion step if one of the eigen value is zero, then pole placement is done at $s = -0.1$. if two eigen values are zero then eigenvalues are placed as complex conjugate pair of $s = -0.1 + 0.0707i$ and $s = -0.1 - 0.0707i$, near to jw axis.

- Following, simulation parameters have been used in this work. Which were calculated form practical setup at Dr. Isenburg's lab.

Table 4.1: Simulation Parameters[13]

a	b	c	d	J_{311}	A	B	c_1	c_2	c_3	c_4
2694.6	6.77	75.5573	7.3916	6.8707	879.6338	22.9022	1	0	10	8

- With most of these small robotic/articulated systems, plat errors due to limited knowledge of the friction model. Whereas, Uncertainty associated with non-ideal sensors. For the CMG, these are quadrature rotary encoders. These sensors will have uncertainty due to quantization giving rise to a uniform noise distribution. Mostly, noise covariances are computed after running experiments on the setup but, in this work following covariances is used to investigate the filter performance, any change in this values can affect the output response.

$$Q = 0.01 \times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

$$R = 0.02 \tag{4.11}$$

- The Initial condition of the state,

$$x_0 = \begin{bmatrix} -\frac{\pi}{3} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.12}$$

- let's say, we want to control the angular position θ_1 i.e. state x_1, then considering it as output y,

$$y = x_1 = \theta_1 \tag{4.13}$$

- Initial guess of error covariance,

$$P_0 = P(0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

- For, calculation of λ in Unscented Kalman filter,

$$\alpha = 5 \tag{4.15}$$

$$\beta = 0 \tag{4.16}$$

$$\kappa = 3 - n \tag{4.17}$$

  Where, n is number of states.

- Weights are calculated based on above λ values.

- Controller poles are placed at 0.2 times of the observer poles.

CHAPTER 5: CONCLUSION

5.1 Discussion

Using Parameters mentioned in section 4.3, following simulation results were obtained.

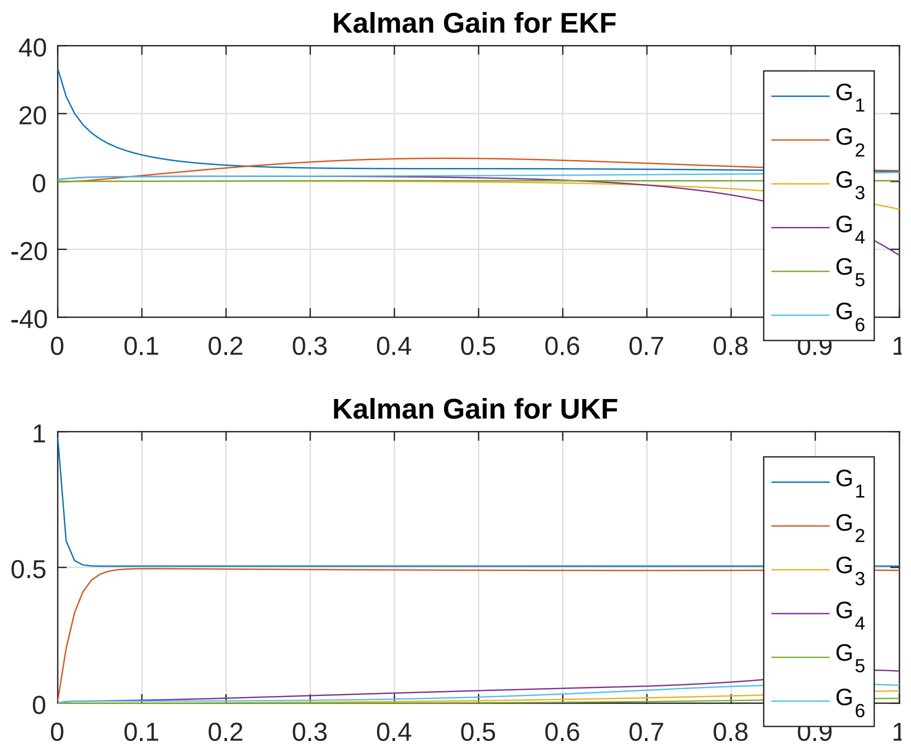

Figure 5.1: Comparison of Kalman Filter Gain

Figure 5.1 compares the kalman gain values for both the filter designed in MATLAB. It is evident from the above figure that EKF converges better then UKF for simulation time of 1 second. In case of UKF, Kalman gain value G_4 doesn't seems to be converging well. Further investigation has been done by changing simulation parameters like initial condition, noise covariances and simulation time.

Figure 5.2 compares the estimated output state i.e. $y = x_1$, it is evident that EKF

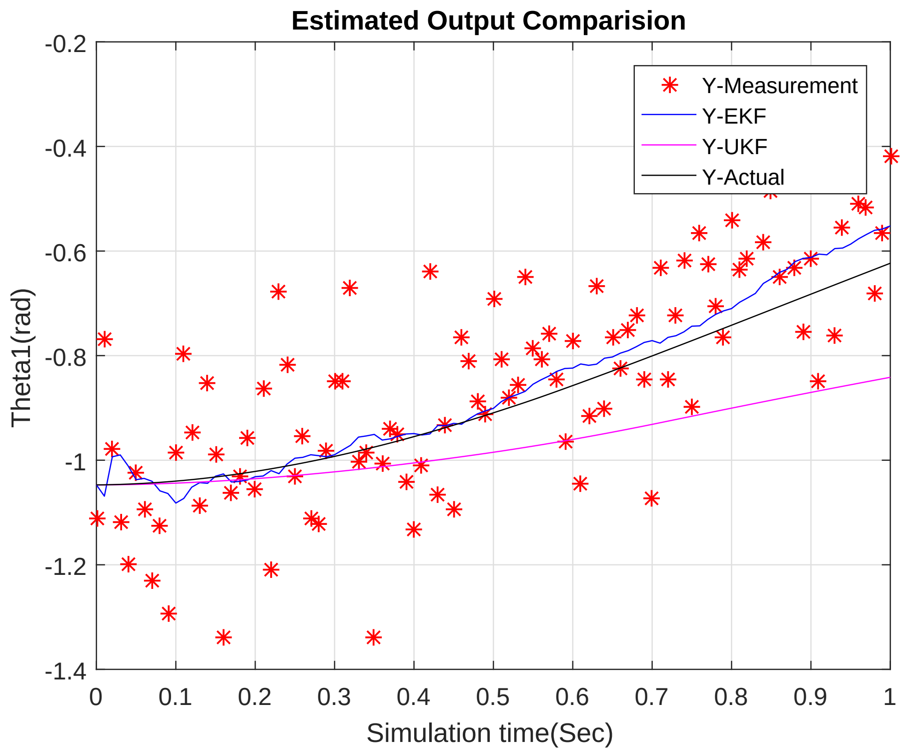

Figure 5.2: Filter Ouput Comparison

follows the true(actual) trajectory well, whereas UKF diverges from its path.

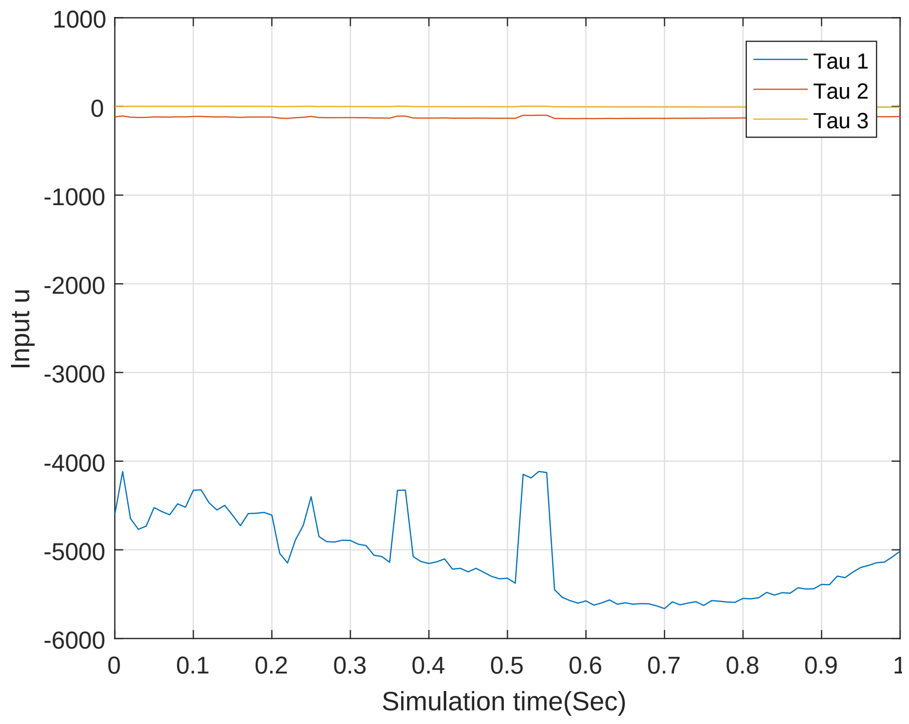

Figure 5.3: Calculated Controller Inputs

Figure 5.3 shows the torque input which being computed by EKF feedback control law as mentioned in equation 2.32

Table 5.1: RMSE Results

EKF_{RMSE}	UKF_{RMSE}
0.0325	0.1185

Above results changes with each run of the code. One of the possible reason can be simulation of random noise using random number generator (rand).

5.1.1 Effect of Initial Condition

Initial condition was changed to following value,

$$x_0 = \begin{bmatrix} \frac{\pi}{3} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.1}$$

Table 5.2: RMSE Results for initial condition $[\pi/3; 0; 0; 0; 0; 0]$

EKF_{RMSE}	UKF_{RMSE}
0.0202	0.1205

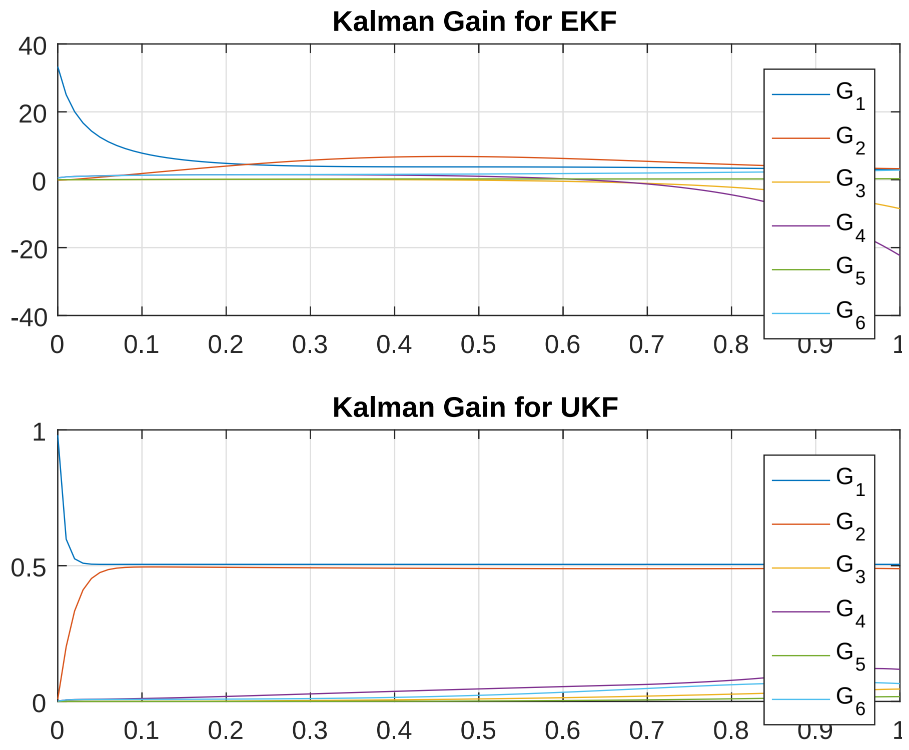

Figure 5.4: Comparison of Kalman Filter Gain for initial condition $[\pi/3; 0; 0; 0; 0; 0]$

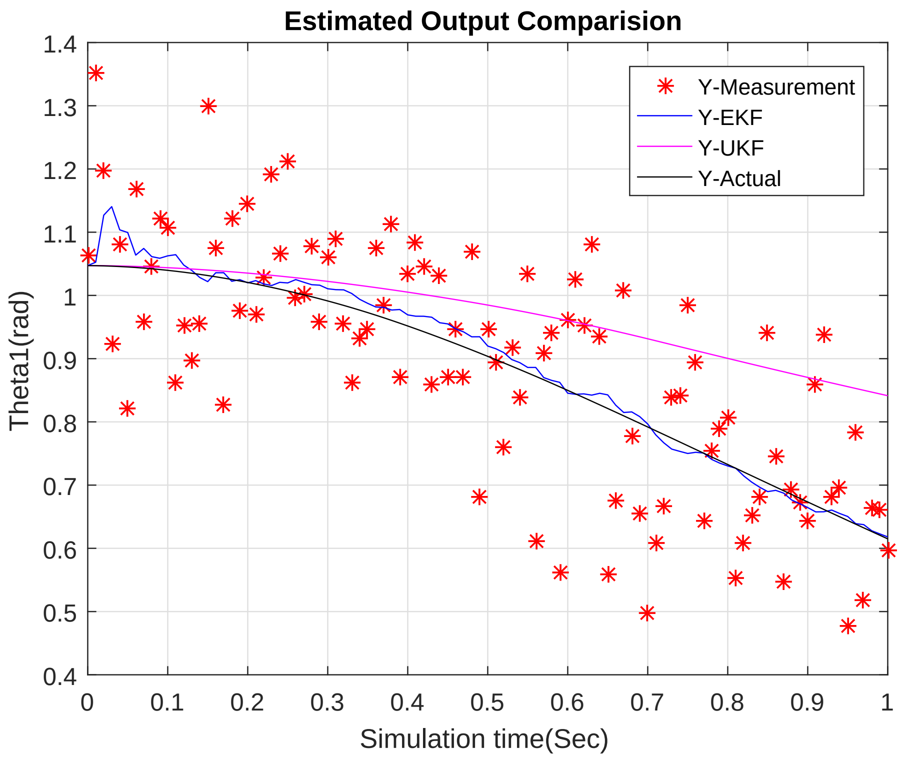

Figure 5.5: Filter Ouput Comparison for initial condition $[\pi/3; 0; 0; 0; 0; 0]$

5.1.2 Effect of Noise Covariances

Noise covariances was changed to 10 times of the previous value i.e.,

$$Q = 0.1 \times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

$$R = 0.2 \tag{5.3}$$

Table 5.3: RMSE Results for 10 times Noise Covariances

EKF_{RMSE}	UKF_{RMSE}
0.0297	0.1157

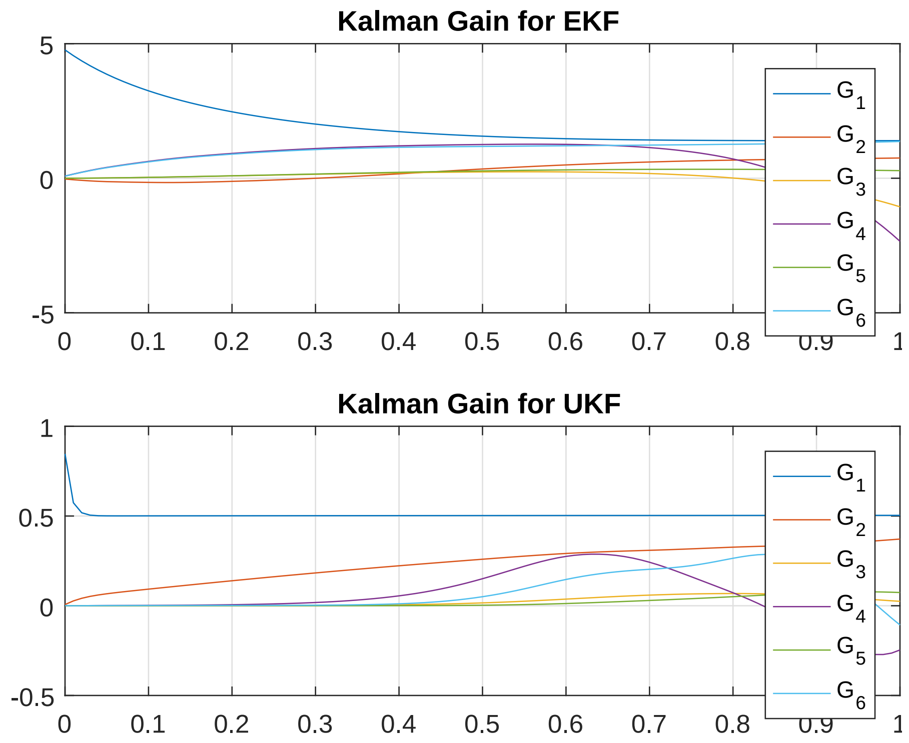

Figure 5.6: Comparison of Kalman Filter Gain for 10 times Noise Covariances

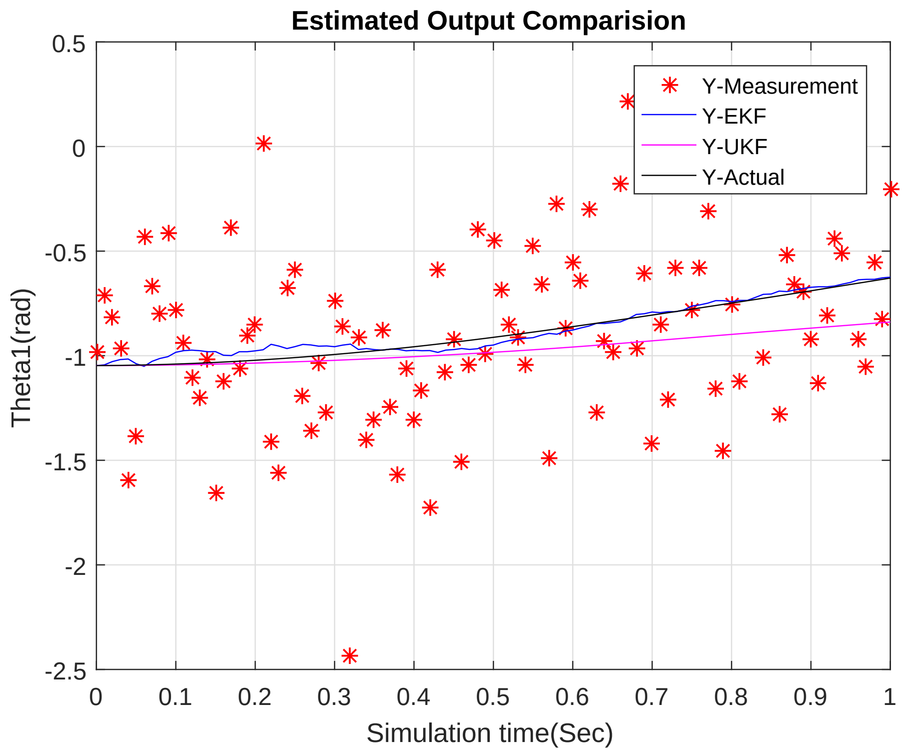

Figure 5.7: Filter Ouput Comparison for for 10 times Noise Covariances

5.1.3 Effect of Simulation Time

Simulation time was changed to 2 second, Noise and initial condition was kept to previous guess.

Table 5.4: RMSE Results for original noise covariances and simulation time 2 sec

EKF_{RMSE}	UKF_{RMSE}
0.0302	0.2367

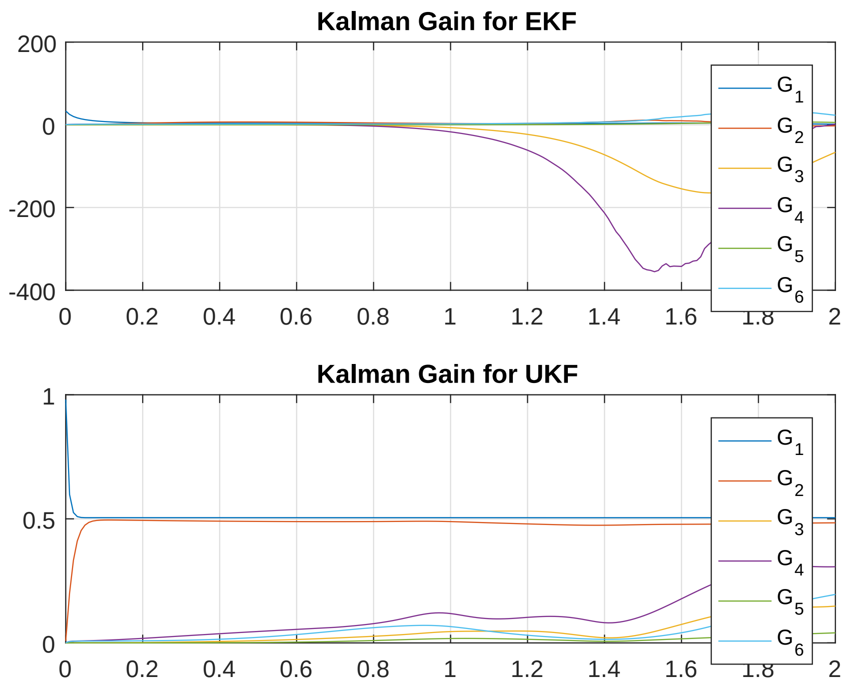

Figure 5.8: Comparison of Kalman Filter Gain for original noise covariances and simulation time 2 sec

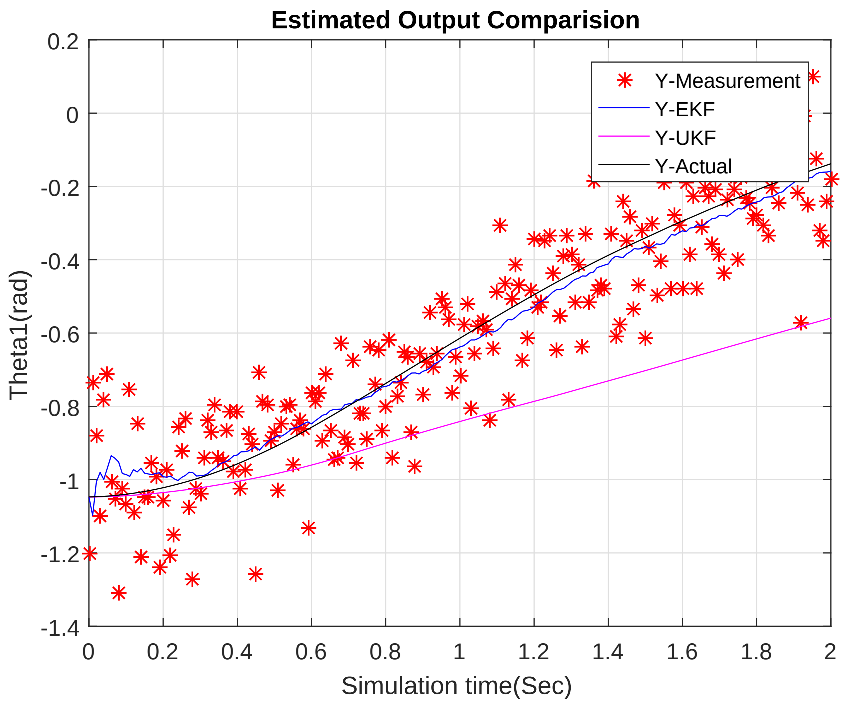

Figure 5.9: Filter Ouput Comparison for for original noise covariances and simulation time 2 sec

5.2 Key Observations

- Simulation results are depended on random number generator, every time we can observe different results, and hence different RMSE values each time. Most of the time it is evident that EKF has less RMSE than UKF for this problem.

- According to above results, one can comment that EKF works reasonably well for this problem, although the big question is the performance of UKF is not so good for this problem.

- For longer time duration (greater than 2 sec), MATLAB's ODE15s solver can not meet the relative tolerances of $1e-2$, and it becomes hard to compute the solution to differential equation, so most of the investigation is done using 2 sec of simulation time.

- The calculated input torque values for τ_1 and τ_2 remains nearly equal to 0, most of the time, where as τ_3 changes polarity based on initial condition.

- Changing the initial condition of x_1 to $+\pi/3$ value, results in similar conclusion of EKF being converging well irrespective of initial condition.
- Increasing noise covariances results in some more variation in Kalman gain, and both the filter doesn't seem to be converging to steady state values, although EKF tracks the true trajectory reasonably well.
- As mentioned earlier, increasing simulation time to 2 sec results in undesirable behavior of kalman gains for both the filter.

5.3 Summary

This research project investigated the dynamics of CMG inverted pendulum for nonlinear state estimation using EKF and UKF. Both algorithms are implemented in MATLAB and results are compared. The result shows that EKF performs well w.r.t UKF, in terms of convergence of the algorithm. It is observed that CMG nonlinear system have stiff non-linearities in dynamics, hence $ode15s$ was used to solve differential equation instead of $ode45$. Although one can explore other solvers and compare the results.

5.4 Future Work

Following recommendation can be made for future work based on above discussion.

- Further investigation on convergence of UKF can be done, here UKF doesn't provide expected results.
- EKF for higher simulation time, has severe problem with convergence to the true state, Also been reported in literature that EKF for highly non-linear system tends to produce undesirable results. Further investigation can be done to properly reason this for CMG problem.

- Discretization of plant can be done and discrete time derivatives of EKF and UKF can be implemented.
- One can investigate on placing controller poles at different location or can use LQG control.
- Comparison on these estimation methods with other techniques like particle filter can be done as a part of the future work.

Appendices

APPENDIX A: MAIN-ROUTINE

mainCMGInvertedPendulum.m

```
%% ----------------------------------------------------
% Main Routine for CMG Computation of EKF and UKF
% ----------------------------------------------------
clearVals(); % clearing workspace
[f,x,h,w,v,f_jac,L,C,M] = symbolicDynamics(); % Symbolic dynamic math
[nState,t,dt,n,x0,Q,R,W,V,P0] = stateInit(); % States and simulation paramters
    initialization

% prefix 'e' stands for extendedKF and 'u' stands for UnscentedKF
% Extended Kalman - Bucy Filter
[eG,eP,eXn,eYn,eXest,eYest,eXa,eYa,eu] = extendedKalmanFilter(Q,R,W,V,x0,P0,t,dt);
extendedKFRMSE = sqrt(mean((eYest - eYa).^2));

% Unscented Kalman - Bucy Filter
[uG,uP,uXest,uYest,uXa,uYa,uu] = unscentedKalmanFilter(Q,R,x0,P0,t,dt);
unscentedKFRMSE = sqrt(mean((uYest - uYa).^2));

% Ploting Kalman Gain values for EKF
figure(1);subplot(2,1,1);plot(t,eG);
legend('G_1','G_2','G_3','G_4','G_5','G_6');grid;title('Kalman Gain for EKF');

% Ploting Kalman Gain values for UKF
figure(1);subplot(2,1,2);plot(t,uG);
legend('G_1','G_2','G_3','G_4','G_5','G_6');grid;title('Kalman Gain for UKF');print -
    depsc epsFig1;

% Ploting Output plot
figure(2);plot(t,eYn,'r*');hold on; plot(t,eYest,'b');plot(t,uYest,'m');plot(t,eYa,'
    black');
legend('Y-Measurement','Y-EKF','Y-UKF','Y-Actual');grid;
xlabel('Simulation time(Sec)'); ylabel('Theta1(rad)');title('Estimated Output
    Comparision');print -depsc epsFig2;

% Plotting Calcuated Controller output
figure(3);plot(t,eu(2:end,:));
legend('Tau 1','Tau 2','Tau 3');grid;
xlabel('Simulation time(Sec)');ylabel('Input u');print -depsc epsFig4;
```

```

% Print RMSE values in Command window
evalin('base','extendedKFRMSE');
evalin('base','unscentedKFRMSE');
```

APPENDIX B: SUB-ROUTINE

Function for jacobian calculation using Symbolic Math Toolbox.

symbolicDynamics.m

```
function [f,x,h,w,v,f_jac,L,C,M,B] = symbolicDynamics()
%% Function to calculate dynamics and jacobians in Sybmolic math
% ---------------------------
syms x1 x2 x3 x4 x5 x6 a b c d g A B J311 u1 u2 u3 w1 w2 w3 w4 w5 w6 v1 v2 v3 v4 v5
    v6;
warning('off','symbolic:sym:sym:DeprecateExpressions')
sym f(x1,x2,x3,x4,x5,x6,a,b,c,d,g,A,B,J311,u1,u2,u3,w1,w2,w3,w4,w5,w6);
sym h(x1,x2,x3,x4,x5,x6,v1,v2,v3,v4,v5,v6);

% ---------------------------
% Defining State Space
% ---------------------------
% x1 = theta1
% x2 = theta1_dot = x1_dot
% x3 = theta2
% x4 = theta2_dot = x3_dot
% x5 = theta3
% x6 = theta3_dot = x5_dot
% ---------------------------
% u1 = tau1
% u1 = tau2
% u1 = tau3
% ---------------------------
H_gama = [ a+b*(sin(x3).^2), c*cos(x3), J311*cos(x3);
           c*cos(x3),       d,          0;
           J311*cos(x3),    0,          J311;];

G_gama = [ A*g*sin(x1)+ B*g*cos(x1)*sin(x3);
           B*g*sin(x1)*cos(x3);
           0;];

D_gama = [ 2*b*x2*x4*cos(x3)*sin(x3) + J311*x4*x6*sin(x3) - c*x4^2*sin(x3);
          -b*x2^2*cos(x3)*sin(x3) + J311*x2*x6*sin(x3);
          -J311*x2*x4*sin(x3);];

```

```
u = [u1;u2;u3];

th_double_dot = H_gama\(u-D_gama-G_gama);

x1_dot = x2 + w1;       %theta1_dot
x2_dot = subs(th_double_dot(1)+w2,[a,b,c,d,g,J311,A,B],...
        [2694.6,6.7728,75.5573,7.3916,9.81,6.8707,879.6338,22.9022]);
x3_dot = x4 + w3;       %theta2_dot
x4_dot = subs(th_double_dot(2)+w4,[a,b,c,d,g,J311,A,B],...
        [2694.6,6.7728,75.5573,7.3916,9.81,6.8707,879.6338,22.9022]);
x5_dot = x6 + w5;       %theta3_dot
x6_dot = subs(th_double_dot(3)+w6,[a,b,c,d,g,J311,A,B],...
        [2694.6,6.7728,75.5573,7.3916,9.81,6.8707,879.6338,22.9022]);

% ------------------------
% x_dot = f(x,u,w) Matrix
% ------------------------
x_dot = [x1_dot; x2_dot; x3_dot; x4_dot; x5_dot; x6_dot;];
x_dot = vpa(x_dot);
x = [x1;x2;x3;x4;x5;x6];
w = [w1;w2;w3;w4;w5;w6];
f = x_dot;
%------------------------
% y = h(x,v) Matrix
%------------------------
v = v1;
y = x1 + v1;
h = y;
f_jac = jacobian(f,x);
fjac = vpa(f_jac);
L = double(jacobian(f,w));
C = double(jacobian(h,x));
M = double(jacobian(h,v));
B = jacobian(f,u);
end
```

Function for clearing workspace and closing all the open figures.

clearVals.m

```matlab
function clearVals()
clc;
evalin('base','clearvars');
evalin('base','close all');
end
```

Function for common initialization.

stateInit.m

```matlab
function [nState,t,dt,n,x0,Q,R,W,V,P0] = stateInit()
%% ------------------
% Initialiization
% ------------------
nState = 6;            %number of states
dt = 0.01;             %ode step size
t = 0:0.01:1;          %time steps
n = length(t);         %number of time steps
Q = 0.1*eye(nState); %Modelling Error or Plant Noise Covariance
R = 0.2 ;             %Measurement Noise Covariance
P0 = eye(nState);      %initial state covraiance
% -------------------------
% Initial guess for x and u
% -------------------------
x0(1) = -pi/3;
x0(2) = 0;
x0(3) = 0;
x0(4) = 0;
x0(5) = 0;
x0(6) = 0;
x0 = x0';
% --------------------------
% Noise Init
% -------------------------
V = sqrt(R)*randn(n,1); %Zero mean gaussian measurement noise data
W = sqrt(Q)*randn(n,6)';
end
```

Function for Extended Kalman Filter

extendedKalmanFilter.m

```
function [G,P,Xm,Ym,X_est,Y_est,Xact,Yact,u] = extendedKalmanFilter(Q,R,W,V,x0,P0,t,
    dt)
%% Main Routine for EKF

% Initialization
Xact(1,:) = x0';
Xm(1,:) = x0';              %State Array Initializationl
X_est(1,:) = x0';           %First Estimated state = initial condition
P(:,:,1) = P0;

% Global Jacobians
L = eye(6);
C = [1 0 0 0 0 0];
M = 1;

% Pre allocating memory to avoid mlint errors and support code-generation
G = zeros(length(t),6);
Ym = zeros(length(t),1);
Y_est = zeros(length(t),1);
Yact = zeros(length(t),1);

Qt = L*Q*L';
Rt = M*R*M';
u = [0 0 0]; % Initial input, since we don't have an estimate

for i = 1:length(t)
    [A,B] = getDerivative(X_est(i,:),u(i,:));
    [P(:,:,i+1),Xm(i+1,:),Ym(i)] = extendedKFPredict(A,C,u(i,:),Qt,Rt,W(:,i),V(i),P
        (:,:,i),X_est(i,:),dt);
    [G(i,:),X_est(i+1,:),Y_est(i)] = extendedKFCorrect(P(:,:,i+1),u(i,:),Rt,C,X_est(i
        ,:),Ym(i),dt);
    [Xact(i+1,:),Yact(i)] = actualTrajac(u(i,:),C,Xact(i,:),dt);
    E = eig((A-G(i,:)'*C))./5;
    if(sum(E(:)==0)==2)
        index = find(E(:)==0);
        E(index(1)) = -0.1 + 0.0707i;
        E(index(2)) = -0.1 - 0.0707i;
    elseif(sum(E(:)==0)==1)
```

```
        index = find(E(:)==0);
        E(index(1)) = -0.1;
    end
    K = place(A,B,E);
    u(i+1,:) = - K*X_est(i+1,:)';
    disp(i);
end
end

function [Pcov,Xp,Yc] = extendedKFPredict(A,C,u,Qt,Rt,w,Vi,P0,X0,dt)
%% Predict Step : Time update

invRt = inv(Rt);
options = odeset('RelTol',1e-2);

% Solve ricatti equation to get covariance at next step
[~,Pval] = ode15s(@(t,P)pRiccati(P,A,C,Qt,invRt),[0 dt],P0,options);

% converting the calculated covariance vector to matrix notation
Pcov = vec2mat(Pval(end,:),6);% Next step covariance

% Project Ahead
[~,stateX] = ode15s(@(t,X)getStates(X,u,w),[0 dt],X0,options); % noisy states
Xp = stateX(end,:); % Predicted noisy state
Yc = X0(1) + Vi; % Current Noisy measurement
end

function dPdt = pRiccati(P,A,C,Qtelda,invRtelda)
%% Riccati Equation for kalman filter
P = reshape(P, size(A)); %Convert from "n^2"-by-1 to "n"-by-"n"
dPdt = A*P + P*A' - P*C.'*invRtelda*C*P + Qtelda; %Determine derivative
dPdt = dPdt(:); %Convert from "n"-by-"n" to "n^2"-by-1
end

function [KalmanGain,Xhatp,Yhatc] = extendedKFCorrect(Pcov,u,Rt,C,X0est,Yc,dt)
%% Correct Step

options = odeset('RelTol',1e-2);

% Calculation for optimal observer gain based on calculated covariance matrix
KalmanGain = (Rt\Pcov*C')';
```

```

% Calculate Estimated States
Yhatc = C*X0est'; % Current Noise less estimate
[~,stateEstX] = ode15s(@(t,X)getStates(X,u,KalmanGain,Yc,Yhatc),[0 dt],X0est,options)
    ;
Xhatp = stateEstX(end,:); % Predicted estimated state;

end

function [Xactp, Yact] = actualTrajac(u,C,X0act,dt)
%% Get Actual states and model response without noise & observer

options = odeset('RelTol',1e-2);

[~,stateActX] = ode15s(@(t,X)getStates(X,u),[0 dt],X0act,options);
Xactp = stateActX(end,:); % Predicted actual state;
Yact = C*X0act'; % Current acutal value
end
```

Function for Unscented Kalman Filter

unscentedKalmanFilter.m

```
function [G,P,x_est,y_est,xact,yact,u] = unscentedKalmanFilter(Q,R,x0,P0,t,dt)
%% Main Routine for Unscented Kalman Filter

% Initialization
xact(1,:) = x0';
x_est(1,:) = x0';            %First Estimated state = initial condition
P(:,:,1) = P0;

% Pre allocating memory to avoid mlint errors and support code-generation
G = zeros(length(t),6);
yact = zeros(length(t),1);
y_est = zeros(length(t),1);

L = eye(6);
C = [1 0 0 0 0 0];
M = 1;

Qt = L*Q*L';
Rt = M*R*M';
u = [0 0 0]; % Initial input, since we don't have an estimate

for i = 1:length(t)
    [A,B] = getDerivative(x_est(i,:),u(i,:));
    y_est(i) = C*x_est(i,:)';
    [x_est(i+1,:),Pp] = unsecntedKFPredict(x_est(i,:),u(i,:),P(:,:,i),Qt,dt);
    [G(i,:),x_est(i+1,:),P(:,:,i+1)] = unscentedKFCorrect(x_est(i+1,:),y_est(i),Pp,Rt
        );
    [xact(i+1,:),yact(i)] = actualTrajac(u(i,:),C,xact(i,:),dt);
    E = eig((A - G(i,:)'*C))./5;
    if(sum(E(:)==0) == 2)            % if there are 2 eigen values which are zero
        index = find(E(:) == 0);
        E(index(1)) = -0.1 + 0.0707i;   % complex conjugate eigen values
        E(index(2)) = -0.1 - 0.0707i;
    elseif(sum(E(:)==0) == 1)        % if there is 1 eigen value which is zero
        index = find(E(:) == 0);
        E(index(1)) = -0.1;             % negentive real values
    end
    K = place(A,B,E);
    u(i+1,:) = - K*x_est(i+1,:)';
    disp(i);
end
end

function [xbarp,Pp] = unsecntedKFPredict(xbar,u,P,Qt,dt)
%% Predict Step / Time Update
xbar = xbar';

% Compute Sigma Points
[Wm,W,c] = getWeights(size(xbar,1));
X = getSigmaPoints(xbar,P,c);

% Propogate each sigma points through function f
options = odeset('RelTol',1e-2);
for i = 1:length(X)
[~,Xodep] = ode15s(@(t,X)getStates(X,u),[0 dt],X(:,i),options);
Xhatp(i,:) = Xodep(end,:);
end

% Compute Predicted mean xbarp
xbarp = Xhatp' * Wm;

% Compute Predicted Covariances
Pp = Xhatp' * W * Xhatp + Qt;
end

function [G,xbar_r,Pp_r] = unscentedKFCorrect(xbarp,y,Pp,Rt)
%% Correct or Update Step / Measurement Update
xbarp = xbarp';

% Compute Sigma Points
[Wm,W,c] = getWeights(size(xbarp,1));
Xp = getSigmaPoints(xbarp,Pp,c);

% Propogate sigma points through function y = h(x) = x1
Yhat = Xp(1,:);

% Compute the Predicted mean
mu = Yhat * Wm;

```

```
% Compute the Predicted Covariances of measurement
Pyy = Yhat * W * Yhat' + Rt;

% Compute cross-covarince of the state and measurment
Pxy = Xp * W * Yhat' ;

% Compute Kalman Gain
G = Pxy/Pyy;

% Compute Filtered state mean and covariances
xbar_r = xbarp + G * (y - mu);
Pp_r = Pp - G * Pyy * G';
end

function SP = getSigmaPoints(X,P0,c)
%% Calculate Sigma Points
A = sqrtm(P0)'; % Square root of Pk-1
SP = [zeros(size(X)) A -A];
SP = sqrt(c)*SP + repmat(X,1,size(SP,2));
end

function [WM,W,c] = getWeights(L)
%% Calculates the weight for Sigma Points
% L is dimention of the problem
alpha = 5;
beta = 0;
kappa = 3 - L;

% Compute the normal weights
lambda = alpha^2 * (L + kappa) - L;
WM = zeros(2*L+1,1);
WC = zeros(2*L+1,1);
for j=1:2*L+1
    if j==1
        wm = lambda / (L + lambda);
        wc = lambda / (L + lambda) + (1 - alpha^2 + beta);
    else
        wm = 1 / (2 * (L + lambda));
        wc = wm;
    end
    WM(j) = wm;
    WC(j) = wc;
end
c = L + lambda;
W = eye(length(WC)) - repmat(WM,1,length(WM));
W = W * diag(WC) * W';
end

function [Xactp, Yact] = actualTrajac(u,C,X0act,dt)
%% Get Actual states and model response without noise & observer

options = odeset('RelTol',1e-2);

[~,stateActX] = ode15s(@(t,X)getStates(X,u),[0 dt],X0act,options);
Xactp = stateActX(end,:); % Predicted actual state;
Yact = C*X0act'; % Current acutal value
end
```

Note : The function getDerivative and getStates are not mentioned in the above code since, they were not fitting in this report's A4 sheet dimension. Please contact author for full copy of the code.

BIBLIOGRAPHY

[1] State Estimation lecture notes- Control and Dynamical Systems (CDS), California Institute of Technology, `https://www.cds.caltech.edu/~murray/wiki/images/b/b3/Stateestim.pdf`

[2] Kirk, Donald E. Optimal control theory: an introduction. Courier Corporation, 2012.

[3] Simon, D., 2006. Optimal state estimation: Kalman, H infinity, and nonlinear approaches. John Wiley & Sons.

[4] Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. Journal of basic Engineering, 82(1), pp.35-45.

[5] Optimization-Based Control lecture notes, Richard M. Murray, Control and Dynamical Systems, California Institute of Technology `http://www.cds.caltech.edu/~murray/books/AM05/pdf/obc-kalman_22Dec09.pdf`

[6] Wan, Eric A., and Ronell Van Der Merwe. ”The unscented Kalman filter for nonlinear estimation.” In Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000, pp. 153-158. Ieee, 2000.

[7] James, Matthew R., and John S. Baras. ”Some General Results and Examples on the Design of Observers for Nonlinear Control Systems.” (1987).

[8] Extended Kalman Bucy Filter, Stengel, Robert F. Optimal control and estimation. Courier Corporation, 2012.

[9] State Estimation with Kalman Filter F. Haugen: Kompendium for Kyb. 2 ved Hgskolen i Oslo. Chapter 102 `http://home.hit.no/~hansha/documents/control/theory/kalmanfilter.pdf`

[10] Stochastic Error Identification and Modeling, Institute of Flight system dynamics,Technical University of Munich webpage `http://www.fsd.mw.tum.de/research/sensors-data-fusion-and-navigation/research-and-competence-areas/data-fusion/`

[11] Wan, Eric A., and Ronell Van Der Merwe., Portland State University Notes `https://www.pdx.edu/biomedical-signal-processing-lab/sites/www.pdx.edu.biomedical-signal-processing-lab/files/ukf.wan_.chapt7_.pdf`

[12] Amengonu, Yawo H., Yogendra P. Kakad, and Douglas R. Isenberg. ”The control moment gyroscope inverted pendulum.” In Advances in Systems Science, pp. 109-118. Springer International Publishing, 2014.

[13] Amengonu, Yawo H. "Geometric methods for control of nonholonomic mechanical systems with applications to the control moment gyroscope and wheeled mobile robots." PhD diss., THE UNIVERSITY OF NORTH CAROLINA AT CHARLOTTE, 2015.