Estimation Theory -ECEN 5523
Reaction Wheel Pendulum Filter Design Project

GIRISH JOSHI

May 6, 2016

1 Reaction Wheel Pendulum: Model

The Reaction Wheel Pendulum, is a simple pendulum with a rotating wheel at the end. The wheel is actuated by a motor mounted on the pendulum. This motor can produce a torque on the wheel, causing the wheel to spin. According to Newtons third law, there is an equal and opposite reaction torque on the pendulum due to conservation of the angular of momentum principle. This reaction torque can be used to control the motion of the pendulum.

The parameters of the Reaction Wheel Pendulum system are given as:

- m_p = Mass of pendulum
- m_r = Mass of rotor
- $m = m_p + m_r$ = Combined mass of rotor and pendulum
- J_p = Moment of inertia of the pendulumabout its center of mass
- J_r = Moment of inertia of the rotor about its center of mass
- l_p = Distance from pivot to the center of mass of the pendulum
- l_r = Distance from pivot to the center of mass of the rotor
- l = Distance from pivot to the center of mass of pendulumand rotor
- θ = Angle of pendulum
- θ_r = Angle of rotor
- $\theta_m = \theta_r - \theta$ = Angle of motor

The equation of motion of Reaction wheel pendulum can be derived from lagrangian principle. The Lagrangian is defined as the difference between the kinetic and potential energy of the system.

$$L(\theta, \dot{\theta}, \theta_r, \dot{\theta}_r) = K.E - P.E \tag{1}$$

for pendulum the lagrangian can be written as,

$$L_p(\theta, \dot{\theta}) = \frac{1}{2} J\theta^2 - mgl(1 - cos(\theta)) \tag{2}$$

and Lagrangian for the rotor can be written as

$$L_r(\theta_r, \dot{\theta}_r) = \frac{1}{2} J_r \theta_r^2 \tag{3}$$

The Lagrange equation of motion, for Reaction wheel pendulum can be written as,

$$\frac{\partial}{\partial t}\left(\frac{\partial L_p}{\partial \dot{\theta}}\right) - \left(\frac{\partial L_p}{\partial \theta}\right) = \tau \tag{4}$$

$$\frac{\partial}{\partial t}\left(\frac{\partial L_r}{\partial \dot{\theta}_r}\right) - \left(\frac{\partial L_r}{\partial \theta_r}\right) = \tau \tag{5}$$

Using expression 2 and 3 in 4 and 5 the equation of motion for the reaction wheeel pendulum can be written as

$$J\ddot{\theta} + mglsin(\theta) = -\tau \tag{6}$$

$$J_r \ddot{\theta}_r = \tau \tag{7}$$

The control torque on the pendulum is generated using the motor connected to the rotor. The torque expression for the motor is given as follows,

$$\tau = k_t i \tag{8}$$

where τ is the reaction torque generated by the motor, K_t is torque constant and i is armature current. The armature current is the function of the voltage applied across the motor coils and back emf generated due to motor coils cutting through the magnetic field of stator magnets.

$$Ri = e - K_b \dot{\theta}_m \tag{9}$$

Using 8 and 9, non-linear equation of motion of reaction wheel pendulum 6 and 7 can be written in state space form as follows.

$$\ddot{\theta} = \frac{-mglsin(\theta)}{g} - \frac{K_t K_b}{JR}\dot{\theta} + \frac{K_t K_b}{JR}\dot{\theta}_r - \frac{K_t e}{JR} \tag{10}$$

$$\ddot{\theta}_r = \frac{K_t K_b}{J_r R}\dot{\theta} - \frac{K_t K_b}{J_r R}\dot{\theta}_r + \frac{K_t e}{J_r R} \tag{11}$$

1.1 Linearization of Equation of Motion

The nonlinear equation of motion for the reaction wheel pendulum 10, 11 can be written as

$$\ddot{\theta} = g_1(\theta, \dot{\theta}, \theta_r, \dot{\theta}_r) \tag{12}$$

$$\ddot{\theta}_r = g_2(\theta, \dot{\theta}, \theta_r, \dot{\theta}_r) \tag{13}$$

Defining the state variable as follows $\theta_1 = \theta$, $\theta_2 = \dot{\theta}$, $\theta_3 = \theta_r$ and $\theta_4 = \dot{\theta}_r$, expression 12 and 13 is written in state space form,

$$\dot{\theta}_1 = f_1(\theta_2) = \theta_2 \tag{14}$$

$$\dot{\theta}_2 = f_2(\theta_1, \theta_2, \theta_3, \theta_4) \tag{15}$$

$$\dot{\theta}_3 = f_3(\theta_4) = \theta_4 \tag{16}$$

$$\dot{\theta}_4 = f_4(\theta_1, \theta_2, \theta_3, \theta_4) \tag{17}$$

Using the first order Taylor series approximation, the state space form of nonlinear equation 14 to 17 can be written in the Linearized form as follows linearized about the equilibrium point $\theta = 0$,

$$\begin{aligned}\dot{X} &= AX + BU \quad (18)\\ Y &= HX + DU \quad (19)\end{aligned}$$

where

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \frac{\partial f_1}{\partial \theta_3} & \frac{\partial f_1}{\partial \theta_4} \\ \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial \theta_2} & \frac{\partial f_2}{\partial \theta_3} & \frac{\partial f_2}{\partial \theta_4} \\ \frac{\partial f_3}{\partial \theta_1} & \frac{\partial f_3}{\partial \theta_2} & \frac{\partial f_3}{\partial \theta_3} & \frac{\partial f_3}{\partial \theta_4} \\ \frac{\partial f_4}{\partial \theta_1} & \frac{\partial f_4}{\partial \theta_2} & \frac{\partial f_4}{\partial \theta_3} & \frac{\partial f_4}{\partial \theta_4} \end{bmatrix} \quad (20)$$

$$B = \begin{bmatrix} \frac{\partial f_1}{\partial e} \\ \frac{\partial f_2}{\partial e} \\ \frac{\partial f_3}{\partial e} \\ \frac{\partial f_4}{\partial e} \end{bmatrix} \quad (21)$$

There by the state space model for reaction wheel pendulum can be written as,

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-mgl}{g} & -\frac{K_t K_b}{JR} & 0 & \frac{K_t K_b}{JR} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{K_t K_b}{JR} & 0 & -\frac{K_t K_b}{JR} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{K_t}{JR} \\ 0 \\ \frac{K_t}{J_r R} \end{bmatrix} e \quad (22)$$

$$Z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} \quad (23)$$

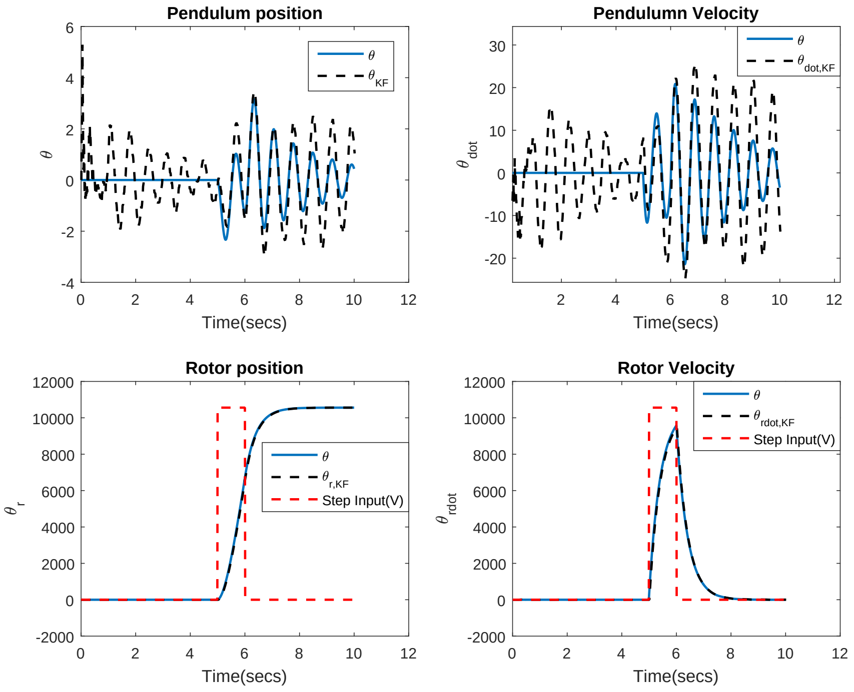

Figure 1: Linear plant model, Simulation with Motor Step Input

2 Angular Sensors

A Precision Hall Effect Angle Sensor "A1332" from *Allegro*^{TM} Microsystems is chosen for the angle measurement of the pendulum, the specification sheet of the sensor is provided in the Annexure. The sensor noise distribution specification at different operating temperature is given following Figure 2. The Sensor measurement Noise Specification (from the Figure2)

	Specification
Measurement Bias	0.3^0
$\pm 3\sigma$Standard Deviation	0.2^0
Variance σ^2	0.0044

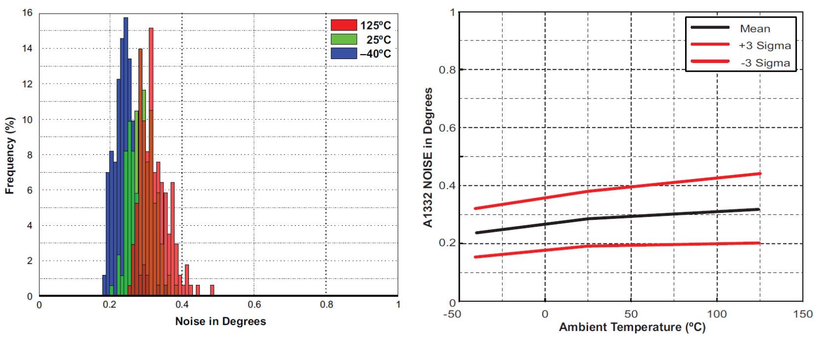

Figure 2: Hall Effect Angle Sensor measurement Noise Distribution vs. operating Temperature

3 Filtering

In this project, the results are presented for estimation of the state vector of the reaction wheel pendulum through Kalman and Extended Kalman filtering. The details of the filter development, filter equation and corresponding results are presented in the subsequent sections.

3.1 Kalman Filtering

In this section details are presented for the Linearized Discrete Kalman Filtering to estimate the state of the Reaction wheel pendulum plant. For Discrete Kalman Filtering(DKF) the linearized plant model needs to be written in the discrete form. The discretization of the linear continues time model for DKF is carried out as follows.
The discrete time model of the system can be written as,

$$X(k+1) = \Phi X(k) + \Psi U(k) \tag{24}$$
$$Z(k+1) = HX(k+1) \tag{25}$$

The value of the state transition matrices Φ,control effective matrix Ψ needs to be determined. While they are independent of the time T and constant, they depend on the value of the sampling interval dt.
The solution of the continues time state equation 18 and 19 for time kT and $(k+1)T$ can be written as

$$X(kT) = e^{AkT}X(0) + e^{AkT}\int_0^{kT} e^{-A\tau}BU(\tau)d\tau \tag{26}$$
$$X((k+1)T) = e^{A(k+1)T}X(0) + e^{A(k+1)T}\int_{kT}^{(k+1)T} e^{-A\tau}BU(\tau)d\tau \tag{27}$$

multiplying 26 by e^{AT} and solving for $e^{A(k+1)T}X(0)$ and further substituting this expression in 27, the expression $X((k+1)T)$ can written as function of $X(kT)$ as follows

$$X((k+1)T) = e^{A(k+1)T}X(k) + \int_{kT}^{(k+1)T} e^{(A(k+1)T-\tau)}BU(kT)d\tau \tag{28}$$

Now we see that as τ ranges from kT to (k + 1)T. Defining a new variable $\lambda = (k+1)T - \tau$. Therefore $d\lambda = -d\tau$ and λ ranges from T to 0 as τ ranges from kT to (k + 1)T. Thus we 28 can be written as

$$X((k+1)T) = e^{AT}X(k) + \int_0^{T} e^{(A\lambda)}BU(kT)d\lambda \tag{29}$$

Therefore equating 24 and 29 we can write discrete system state transition matrix as follows,

$$\Phi = e^{AT} \tag{30}$$
$$\Psi = \int_0^{T} e^{(A\lambda)}d\lambda B \tag{31}$$

where T is sampling time.
With the discrete time system definition the kalman filtering state estimation equation can be written as

Model	Dynamics $\dot{X} = f(x, U, W)$ Measurement Equation $Z(k+1) = HX(k+1)$
Initialization	$\hat{X}(k\|k) = \hat{X}(0)$ Error Covariance $P(k\|k) = E(\tilde{X}(0)\tilde{X}(0)^T)$
State Prediction	$\hat{X}(k+1\|k) = \Phi\hat{X}(k\|k) + \Psi U(k)$
Covarince propagation	$P(k+1\|k) = \Phi P(k\|k)\Phi^T + \Gamma Q \Gamma^T$
Kalman Gain Computation	$K(k+1) = P(k+1\|k)H^T(HP(k+1\|k)H^T + R)^{-1}$
State Update	$\hat{X}(k+1\|k+1) = \hat{X}(k+1\|k) + K(k+1)(Z(k+1) - H\hat{X}(k+1\|k))$
Covariance Update	$P(k+1\|k+1) = (I-KH)P(k+1\|k)(I-KH)^T + KRK^T$ or $P(k+1\|k+1) = (I-KH)P(k+1\|k)$

Assuming the Parameters for reaction wheel pendulum as follows

- $m_p = 0.2164$ kg
- $J_p = 2.233 \times 10^{-4} kgm^2$
- $m_r = 0.0850$kg
- $J_r = 2.495 \times 10^{-5} kgm^2$
- $m = 0.3014$ kg
- $l = 0.1200$ m
- $l_p = 0.1173$ m
- $l_r = 0.1270$ m
- $J = 4.572x10^{-3} kgm^2$

The Linear discretized plant model for reaction wheel pendulum can be written as follows

$$\begin{bmatrix} \theta_1(k+1) \\ \theta_2(k+1) \\ \theta_3(k+1) \\ \theta_4(k+1) \end{bmatrix} = \begin{bmatrix} 0.9961 & 0.0100 & 0 & 0.0000 \\ -0.7750 & 0.9960 & 0 & 0.0001 \\ -0.0000 & 0.0001 & 1.0000 & 0.0099 \\ -0.0096 & 0.0245 & 0 & 0.9754 \end{bmatrix} \begin{bmatrix} \theta_1(k \\ \theta_2(k) \\ \theta_3(k) \\ \theta_4(k) \end{bmatrix} + \begin{bmatrix} -0.0000 \\ -0.0048 \\ 0.0089 \\ 0.8852 \end{bmatrix} e \tag{32}$$

The measurement vector consists encoder measurement of the absolute angle of the pendulum and motor rotation. Therefore the measurement equation is written as

$$Z(k+1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \theta_1(k+1) \\ \theta_2(k+1) \\ \theta_3(k+1) \\ \theta_4(k+1) \end{bmatrix} \tag{33}$$

The Noise covariance matrix R is designed based on the sensor specification,

$$R = \begin{bmatrix} 0.04^0 & 0 \\ 0 & 0.04^0 \end{bmatrix} \tag{34}$$

and process noise covariance Q is tuned for the optimum performance of the filter,

$$Q = \begin{bmatrix} 1e-4 & 0 \\ 0 & 1e-4 \end{bmatrix} \tag{35}$$

3.1.1 Results: Kalman Filtering

Matlab "ODE-45" integration is used for simulating the actual plant model for Reaction Wheel Pendulum. The dt for the simulation is chosen as $\frac{1}{10}^{th}$ of the inverse fo the highest eigen value of the continues time, linear state transition matrix.
the continues time state equation can be written as,

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1.0000 & 0 & 0 \\ -77.6046 & -0.0136 & 0 & 0.0136 \\ 0 & 0 & 0 & 1.0000 \\ 0 & 2.4868 & 0 & -2.4868 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} + \begin{bmatrix} 0 \\ -0.4953 \\ 0 \\ 90.7600 \end{bmatrix} e \tag{36}$$

dt for the simulation is

$$dt = \frac{1}{10 * max(|||[Re(\lambda_i)]|||)} = \frac{1}{10 * 2.5} = 0.04secs$$

The figure 3 shows the kalman filter performance for pendulum initial condition much away from the equilibrium point. The motor input, a pulse of voltage is also applied to the motor at $t = 5secs$. It can be observed that the kalman filter estimates are not accurate enough, due to linearization error as we are operating in the region away from the equilibrium point.

Simulation in figure 4 demonstrates that system can be put into oscillation using external input to motor. The pendulum is at placed at its equilibrium position to start with and motor voltage $V = 1volts$ is applied for $1sec$ starting from $t = 5secs$. Due to the torque generated by the motor the pendulum is set in oscillations. Since the pendulum oscillates in the neighborhood of the equilibrium points, after initial transient error settles, the Linearized Kalman filter estimates does good in state estimation.

Figure 5 simulates the system with no external inputs. The pendulum is initially displaced to very small angle $\theta_1 = 3^0$ and released to freely oscillate under the influence of the gravity. Since the peak oscillation angles are with in the close neighborhood of the equilibrium point, the performance of the "Linearized Kamlan filter" in estimating the state of the pendulum and rotor is good.

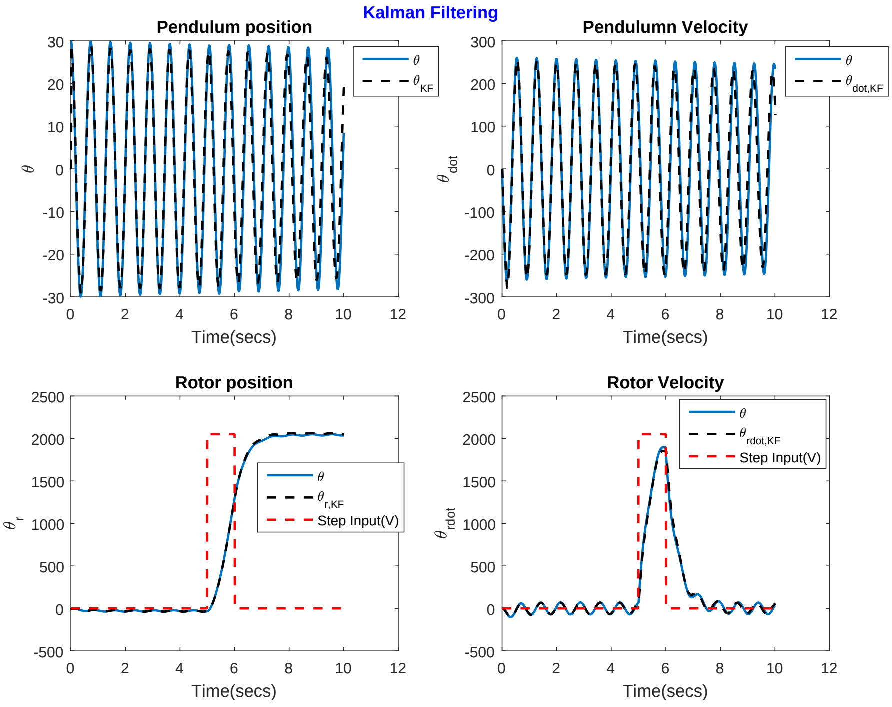

Figure 3: Initial Condition: $\theta_1 = 30^0$, $\theta_2 = 0$ and motor pulse of 5V is applied at $t = 5secs$

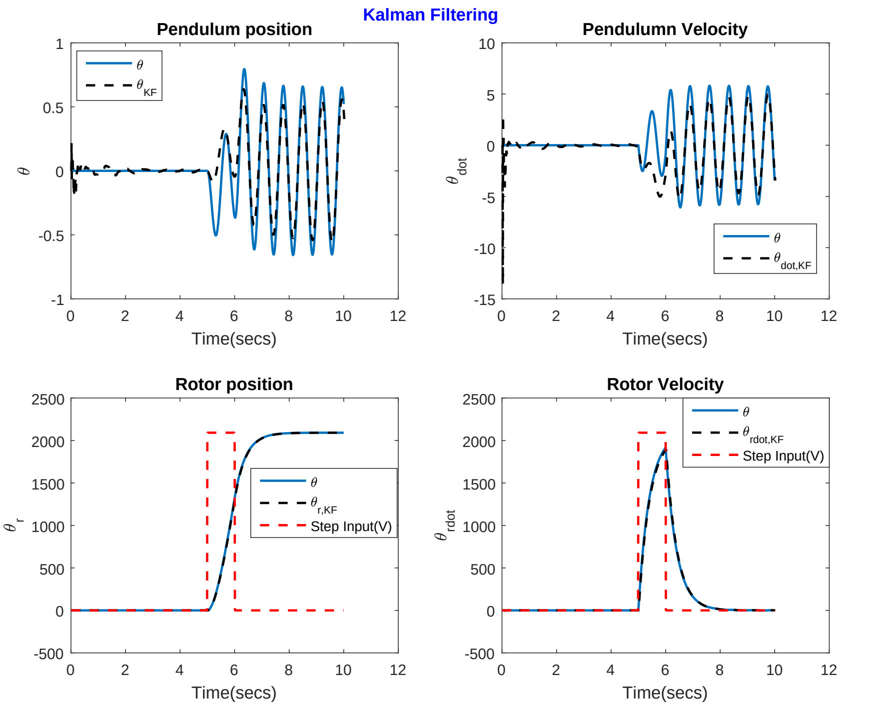

Figure 4: Initial Condition: $\theta_1 = 0$, $\theta_2 = 0$ and motor pulse of 5V is applied at $t = 5secs$

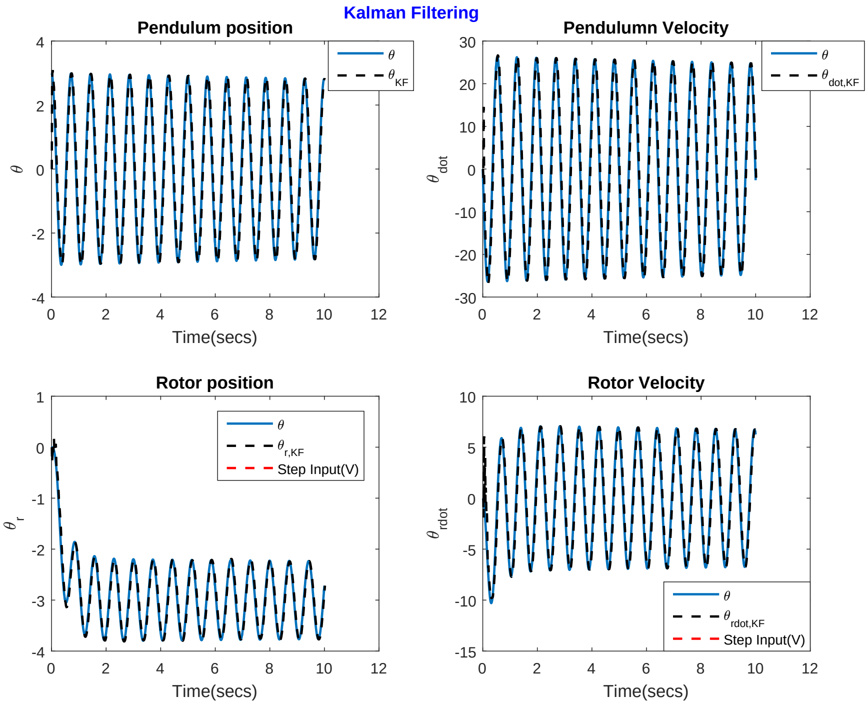

Figure 5: Initial Condition: $\theta_1 = 3^0$, $\theta_2 = 0$ and no motor pulse applied

3.2 Extended Kalman Filtering

The extended Kalman filter (EKF) is the nonlinear version of the Kalman filter which linearizes about an estimate of the current mean and covariance. With nonlinear systems the Gaussian input doest not necessarily produce a Gaussian output (unlike linear case). Fundamentally assumption in EKF is the estimated state $\hat{X}$ is close to true state X at all time, and hence the error dynamics can be fairly accurately represented as the linearized plant dynamics about the estimate $\hat{X}$.

The nonlinear system is linearized, as in 18 and 19 where

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \frac{\partial f_1}{\partial \theta_3} & \frac{\partial f_1}{\partial \theta_4} \\ \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial \theta_2} & \frac{\partial f_2}{\partial \theta_3} & \frac{\partial f_2}{\partial \theta_4} \\ \frac{\partial f_3}{\partial \theta_1} & \frac{\partial f_3}{\partial \theta_2} & \frac{\partial f_3}{\partial \theta_3} & \frac{\partial f_3}{\partial \theta_4} \\ \frac{\partial f_4}{\partial \theta_1} & \frac{\partial f_4}{\partial \theta_2} & \frac{\partial f_4}{\partial \theta_3} & \frac{\partial f_4}{\partial \theta_4} \end{bmatrix}_{x=\hat{X}(k|k)} \tag{37}$$

The above linearized state transition matrix is evaluated at every updated estimate $\hat{X}(k|k)$. Subsequently the discretized system dynamics $\Phi(k+1,k)$ is evaluated at every time step of estimator using the discretization procedure given in 30 and 31.

The Extended-Kalman Filtering is summarized in following table

Model	Dynamics $\dot{X} = f(x, U, W)$ Measurement Equation $Z(k+1) = HX(k+1)$
Initialization	$\hat{X}(k\|k) = \hat{X}(0)$ Error Covariance $P(k\|k) = E(\tilde{X}(0)\tilde{X}(0)^T)$
State Prediction	$\dot{\hat{X}} = f(\hat{X}(k\|k), U(k)$ RK-4 Integration is used to solve the differential equation for predicted state variable $\hat{X}(k+1 \mid k)$
Linearize the system	Evaluate A given in 37 at at $X(k\|k)$ and discretize the system to evaluate $\Phi(k+1, k)$
Covarince propagation	$P(k+1\|k) = \Phi P(k\|k)\Phi^T + \Gamma Q \Gamma^T$
Kalman Gain Computation	$K(k+1) = P(k+1\|k)H^T(HP(k+1\|k)H^T + R)^{-1}$
State Update	$\hat{X}(k+1\|k+1) = \hat{X}(k+1\|k) + K(k+1)(Z(k+1) - H\hat{X}(k+1\|k))$
Covariance Update	$P(k+1\|k+1) = (I - KH)P(k+1\|k)(I - KH)^T + KRK^T$ or $P(k+1\|k+1) = (I - KH)P(k+1\|k)$

3.2.1 Results: Extended Kalman Filtering

Figure 6, shows the simulation results for very large displacement of the pendulum from the equilibrium position. Since the EKF linearizes the plant the EKF estimate of the state are more accurate compared to linearized kalman filter estimate.

Figure 7 and 8 shows the plot with pendulum initial condition in neighborhood of the 0^0 angle with and without the motor impulse. For this case EKF and KF estimates are comparable because the linearization errors are small and KF also performs good compared to EKF in state estimate.

Figure 9 shows the case for initial condition of the pendulum angle to be near upright position thats is $\theta_1 = 150^0$. A well tuned EKF estimates the states very well even when we are operating the system very far away from the equilibrium position. Since the in EKF the system is linearized on the previous updated state estimate the EKF does well in estimating the states where as KF fails.

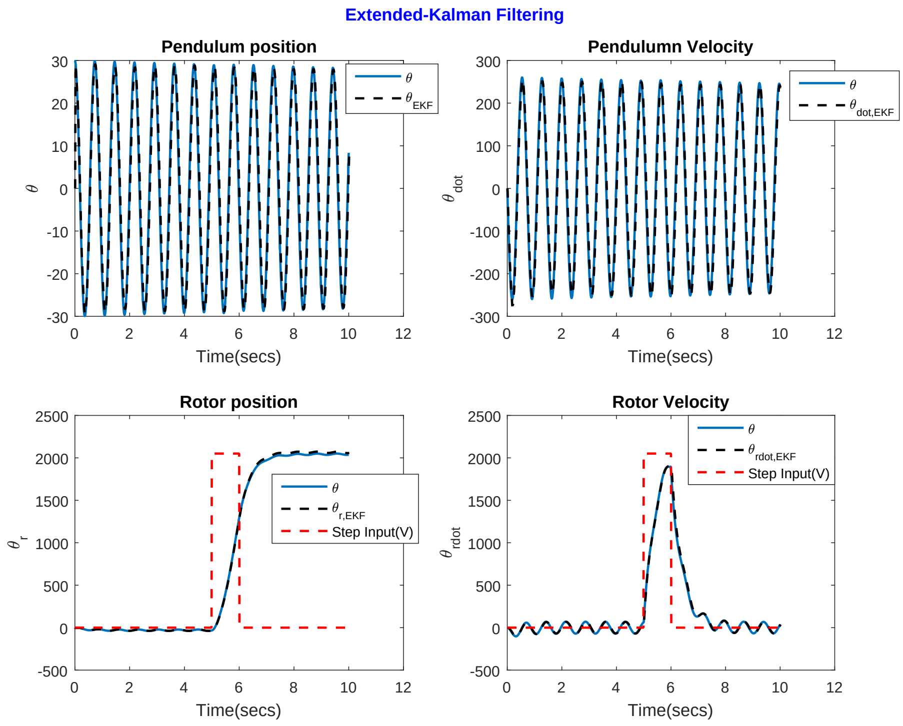

Figure 6: Initial Condition: $\theta_1 = 30^0$, $\theta_2 = 0$ and motor pulse of 5V is applied at $t = 5secs$

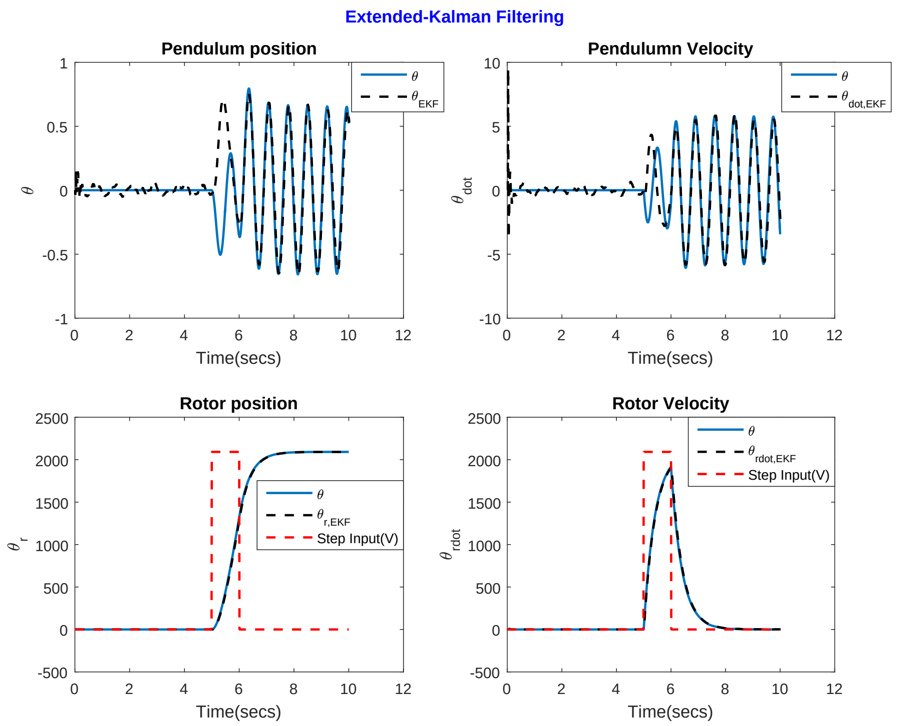

Figure 7: Initial Condition: $\theta_1 = 0$, $\theta_2 = 0$ and motor pulse of 5V is applied at $t = 5secs$

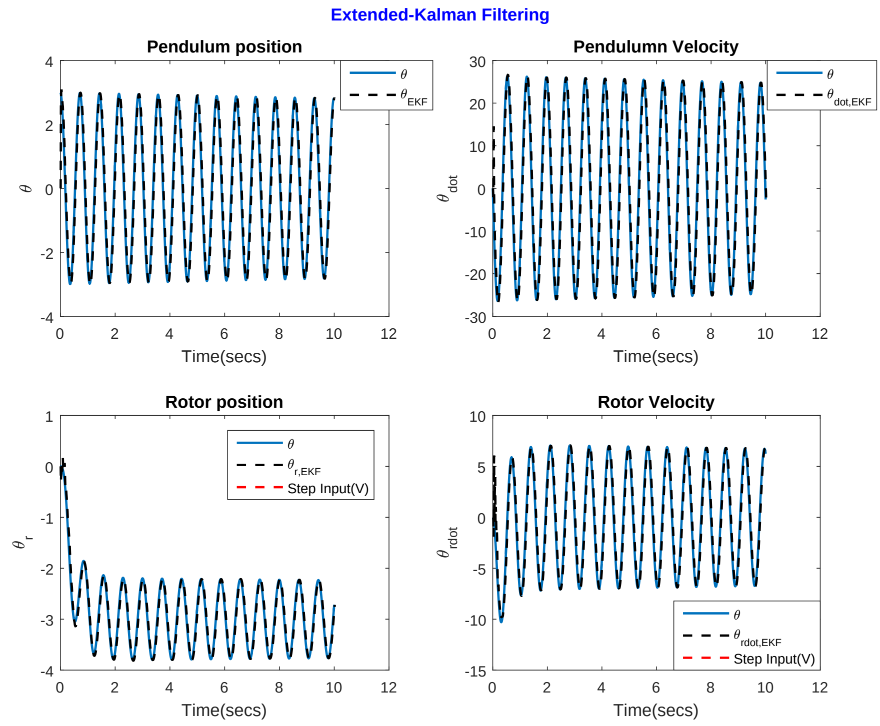

Figure 8: Initial Condition: $\theta_1 = 3^0$, $\theta_2 = 0$ and no motor pulse applied

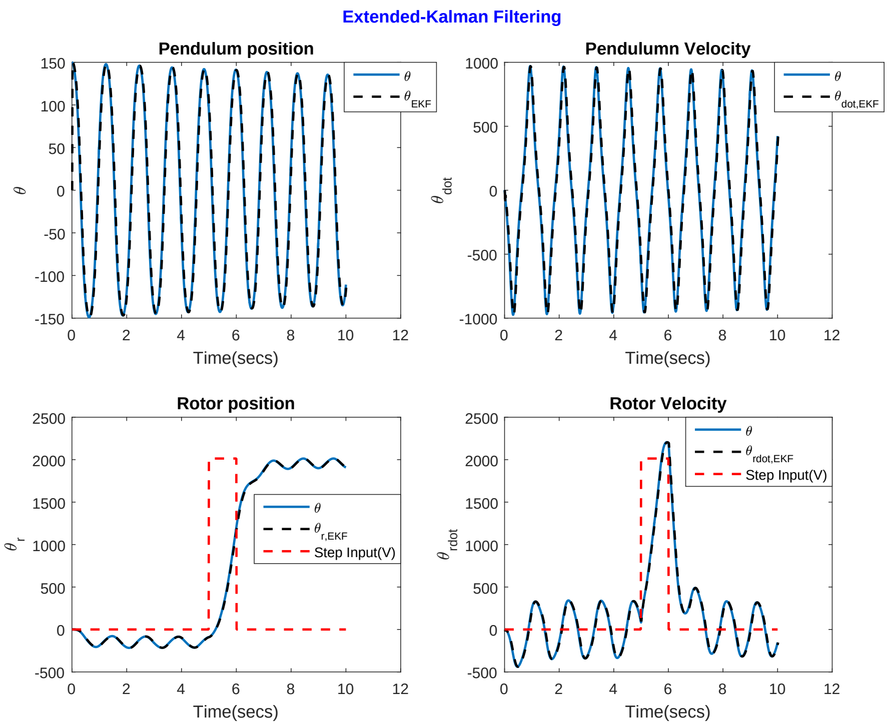

Figure 9: Initial Condition: $\theta_1 = 150^0$, $\theta_2 = 0$ and motor pulse of 5V is applied at $t = 5secs$

4 conclusions

- Fundamental assumption in EKF is that true state $X(t)$ is sufficiently close to estimated state $\hat{X}(t)$, hence the linearized system about $\hat{X}(t)$ is fairly accurate for state propagation.
- Term P_0 and Q are tunable parameters and was found that these value needs to be tuned and one single value of P_0 and Q does not perform optimally for all the initial conditions.
- For initial conditions away from the neighborhood of the stable position, KF performance degrades, where as EKF does well even in the cases where initial conditions for pendulum are near upright positions.
- Tuned Q values for EKF are much smaller compared to what need for good performance of the KF.

5 Matlab-codes

5.1 Kalman and Extended Kalman Estimate main file and Variable Initialization

```

% This program Simulates the Reaction Wheel Pendulum
% 04/27/2016
% Author: Girish Joshi
% Term Project: Estimation Theory
% Mechanical and Aerospace Engineering, Oklahoma State University.

clear all
clc

% Simulation Time step
global dt
dt = 0.01;
k = 1;
Final_time = 10;
Time_Count(1) = 1;
%
Initialization;

% Intial Condition
x = [30*pi/180;0;0;0]; % Continues State Initial Condition
X(:,1) = mvnrnd([0;0;0;0],sqrt(P_updated));
X_ekf(:,1) = mvnrnd([0;0;0;0],sqrt(P_updated_ekf));
Intial_Condition = x;

% STEP INPUT
V_step = 1;
step_start_time = 5;
step_end_time = 6;

```

```
% Intialize vector for Data logging
Time = [];
States = [];
State_Error_kf = [];
State_Error_ekf = [];

[T x] = ode45(@(t,state) ReactionWheel_Pendulum(t,state,V_step,
    step_start_time,step_end_time),[0:dt:Final_time],Intial_Condition);

for t = 0:dt:Final_time

    % Motor Voltage Pulse
    if t >= step_start_time && t<= step_end_time

        V= V_step;

    else
        V = 0;
    end

    %%%%%%%%%%%%%%%%% KALMAN FILTER %%%%%%%%%%%%%%%%%%%%%%%%%%%

     % PREDICTION STEP

     X(:,k+1) = A*X(:,k) + B*V;

     % MEASUREMENT VECTOR

     Z(:,k+1) = H*x(k,:)' + [random('norm',0,sqrt(R_noise(1,1)));random
         ('norm',0,sqrt(R_noise(2,2)))];

     % Error Covariance PREDICTION

      P_predictor = A*P_updated*A' + G*Q_kf*G';

     % Kalman Gain

      Kalman_Gain = P_predictor*H'*inv(H*P_predictor*H' + R_noise);

     % Correction Step

     X(:,k+1) = X(:,k+1) + Kalman_Gain*(Z(:,k+1) - H*X(:,k+1));

    % Error Covariance Propagation

      P_updated = (eye(4) - Kalman_Gain*H)*P_predictor*(eye(4) -
          Kalman_Gain*H)' + Kalman_Gain*R_noise*Kalman_Gain';

```

```
        %%

        %%%%%%%%% Extended Kalman Filtering %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        %PREDICTION STEP

        X_ekf(:,k+1) = Reaction_Wheel_Pendulum(X_ekf(:,k),V);

        % PHI linearized at X(k|k)

        PHI = STATE_TRANSITION_JACOBIAN(X_ekf(:,k));

        % Error Covariance PREDICTION

        P_predictor_ekf = PHI*P_updated_ekf*PHI'+ G*Q_ekf*G';

         % Kalman Gain

        Kalman_Gain_ekf = P_predictor_ekf*H'*inv(H*P_predictor_ekf*H' + R_noise);

        % Correction Step

        X_ekf(:,k+1) = X_ekf(:,k+1) + Kalman_Gain_ekf*(Z(:,k+1) - H*X_ekf(:,k+1));

        % Error Covariance Propagation

        P_updated_ekf = (eye(4) - Kalman_Gain_ekf*H)*P_predictor_ekf*(eye(4) - Kalman_Gain_ekf*H)'+ Kalman_Gain_ekf*R_noise*Kalman_Gain_ekf';

       % STORING DATA FOR PLOTTING %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

       P_PRED(:,k) = diag(P_predictor);
       P_UPDATE(:,k) = diag(P_updated);
       P_PRED_ekf(:,k) = diag(P_predictor_ekf);
       P_UPDATE_ekf(:,k) = diag(P_updated_ekf);
       Time = [Time,t];
       Time_Count(k+1) = k+1;
       State_Error_kf = [State_Error_kf,x(k,:)'-X(:,k+1)];
       State_Error_ekf = [State_Error_ekf,x(k,:)'-X_ekf(:,k)];
       Voltage(k) = V;
       k = k+1;

end
Plots;
```

```
global mp Jp mr Jr m lp l lr J g

% Pendulum Parameters
mp = 0.2164; % Mass of Pendulum (kg)
Jp = 2.333e-4; % moment of inertia of the pendulum about its center of mass (Kgm2)
mr = 0.0850; % mass of the wheel
Jr = 2.495e-5; % moment of inertia of the rotor about its center of mass
m  = 0.3014; % combined mass of rotor and pendulum
lp = 0.1173; % distance from pivot to the center of mass of the pendulum
l  = 0.12; % distance from pivot to the center of mass of pendulumand rotor
lr = 0.1270; % distance from pivot to the center of mass of the rotor
J  = 4.572e-3;  % moment of inertia of the combined system
g = 9.81; % Gravitational Constant

% Motor Parameters
global Kt R Kb dt

Kt = 27.4e-3; % Current Constant for Torque(Nm/A)
R  = 12.1; % Armatuer Resistance (ohms)
Kb = 27.4e-3; % Back EMF Constants

% Continues time State Matrices
global A_cont B_cont H_cont D_cont

A_cont  = [0 1 0 0
           -(m*g*l)/J -Kb*Kt/(J*R) 0 Kb*Kt/(J*R)
           0 0 0 1
           0 Kb*Kt/(Jr*R) 0 -Kb*Kt/(Jr*R)];

B_cont = [0;-Kt/(J*R);0;Kt/(Jr*R)];

H_cont = [1 0 0 0;-1 0 1 0];

D_cont = 0;

% Discrete Time State matrices

global A B H D
Ts = dt;
ssmodel = ss(A_cont,B_cont,H_cont,D_cont);
ssmodel_discete = c2d(ssmodel, Ts, 'forward');

A = ssmodel_discete.a;
B = ssmodel_discete.b;
```

```
H = ssmodel_discete.c;
D = ssmodel_discete.d;
G = [0;1;0;1];

% Kalman Variables

P_updated = 0.05*eye(4);
Q_kf = 0.01;
Q_ekf = 0.001;
R_noise = [1.2185e-02 0;0 1.2185e-02];
P_updated_ekf = 0.05*eye(4);
```

5.2 Function File for ODE45 Integration for True Plant Model Propagation

```
function [x_dot] = ReactionWheel_Pendulum(t,x,e_step,step_t1,step_t2)
        global mp Jp mr Jr m lp l lr J g
        global Kt R Kb

        if t >= step_t1 && t<=step_t2
            e = e_step;
        else
            e = 0;
        end

        theta1_Dot = x(2);
        theta2_Dot = -(m*g*l*sin(x(1)))/J + Kt*Kb*x(4)/(J*R) - Kt*Kb*x
           (2)/(J*R)-Kt*e/(J*R);

        thetar1_Dot = x(4);
        thetar2_Dot = -Kt*Kb*x(4)/(Jr*R) + Kt*Kb*x(2)/(Jr*R) + Kt*e/(Jr
           *R);

        x_dot=[theta1_Dot;theta2_Dot;thetar1_Dot;thetar2_Dot];

    end
```

5.3 Function File for RK-4 Integration for Prediction step of the state

```
function [x] = Reaction_Wheel_Pendulum(x,V)

   global dt


   xp_actual=Actual_State_Model(x,V);

   % 1st step of RK4
   rk1=dt*xp_actual;
   x1=x+rk1/2;

```

```
    % 2nd step of RK4
    xp_actual=Actual_State_Model(x1,V);
    rk2=dt*xp_actual;
    x1=x+rk2/2;

    % 3rd step of RK4
    xp_actual=Actual_State_Model(x1,V);
    rk3=dt*xp_actual;
    x1=x+rk3;

    % 4th step of RK4
    xp_actual=Actual_State_Model(x1,V);
    rk4=dt*xp_actual;

    x = x+(rk1+2.0*(rk2+rk3)+rk4)/6;


%% STATE Model

     function [x_dot] = Actual_State_Model(X,e)

          global mp Jp mr Jr m lp l lr J g
          global Kt R Kb

          theta1_Dot = X(2);
          theta2_Dot = -(m*g*l*sin(X(1)))/J + Kt*Kb*X(4)/(J*R) - Kt*Kb*X
             (2)/(J*R)-Kt*e/(J*R);

          thetar1_Dot = X(4);
          thetar2_Dot = -Kt*Kb*X(4)/(Jr*R) + Kt*Kb*X(2)/(Jr*R) + Kt*e/(Jr
             *R);

          x_dot=[theta1_Dot;theta2_Dot;thetar1_Dot;thetar2_Dot];

     end

end
```

5.4 Function File Results and Plots

```
figure(1)
subtitle('Kalman');
subplot(2,2,1)
plot(Time,x(:,1)*180/pi,'Linewidth',1.5)
hold on
plot(Time_Count*dt,X(1,:)*180/pi,'k--','Linewidth',1.5);
legend('\theta','\theta_{KF}')
title('Pendulum position','fontsize',15)
```

```
set(gca,'fontsize',10)
xlabel('Time(secs)');
ylabel('\theta');
p=mtit('Kalman Filtering',...
                 'fontsize',20,'color',[0 0 1],...
                 'xoff',0.6,'yoff',.05);

subplot(2,2,2)
plot(Time,x(:,2)*180/pi,'Linewidth',1.5);
hold on
plot(Time_Count*dt,X(2,:)*180/pi,'k--','Linewidth',1.5);
legend('\theta','\theta_{dot,KF}')
title('Pendulumn Velocity','fontsize',15);
set(gca,'fontsize',10)
xlabel('Time(secs)');
ylabel('\theta_{dot}');

subplot(2,2,3)
plot(Time,x(:,3)*180/pi,'Linewidth',1.5);
hold on
plot(Time_Count*dt,X(3,:)*180/pi,'k--','Linewidth',1.5);
hold on
plot(Time,Voltage*max(x(:,3))*180/(pi*max(Voltage)),'r--','Linewidth'
    ,1.5);
legend('\theta','\theta_{r,KF}','Step Input(V)')
title('Rotor position','fontsize',15)
set(gca,'fontsize',10)
xlabel('Time(secs)');
ylabel('\theta_r');

subplot(2,2,4)
plot(Time,x(:,4)*180/pi,'Linewidth',1.5);
hold on
plot(Time_Count*dt,X(4,:)*180/pi,'k--','Linewidth',1.5);
hold on
plot(Time,Voltage*max(x(:,3))*180/(pi*max(Voltage)),'r--','Linewidth'
    ,1.5);
legend('\theta','\theta_{rdot,KF}','Step Input(V)')
title('Rotor Velocity','fontsize',15);
set(gca,'fontsize',10)
xlabel('Time(secs)');
ylabel('\theta_{rdot}');


figure(2)
subplot(2,2,1)
plot(Time,x(:,1)*180/pi,'Linewidth',1.5)
hold on
```

```
plot(Time_Count*dt,X_ekf(1,:)*180/pi,'k--','Linewidth',1.5);
legend('\theta','\theta_{EKF}')
title('Pendulum position','fontsize',15)
set(gca,'fontsize',10)
xlabel('Time(secs)');
ylabel('\theta');

subplot(2,2,2)
plot(Time,x(:,2)*180/pi,'Linewidth',1.5);
hold on;
plot(Time_Count*dt,X_ekf(2,:)*180/pi,'k--','Linewidth',1.5);
legend('\theta','\theta_{dot,EKF}')
title('Pendulumn Velocity','fontsize',15);
set(gca,'fontsize',10)
xlabel('Time(secs)');
ylabel('\theta_{dot}');

subplot(2,2,3)
plot(Time,x(:,3)*180/pi,'Linewidth',1.5);
hold on
plot(Time_Count*dt,X_ekf(3,:)*180/pi,'k--','Linewidth',1.5);
hold on
plot(Time,Voltage*max(x(:,3))*180/(pi*max(Voltage)),'r--','Linewidth'
    ,1.5);
legend('\theta','\theta_{r,EKF}','Step Input(V)')
title('Rotor position','fontsize',15)
set(gca,'fontsize',10)
xlabel('Time(secs)');
ylabel('\theta_r');

subplot(2,2,4)
plot(Time,x(:,4)*180/pi,'Linewidth',1.5);
hold on
plot(Time_Count*dt,X_ekf(4,:)*180/pi,'k--','Linewidth',1.5);
hold on
plot(Time,Voltage*max(x(:,3))*180/(pi*max(Voltage)),'r--','Linewidth'
    ,1.5);
legend('\theta','\theta_{rdot,EKF}','Step Input(V)')
title('Rotor Velocity','fontsize',15);
set(gca,'fontsize',10)
xlabel('Time(secs)');
ylabel('\theta_{rdot}');
q=mtit('Extended-Kalman Filtering',...
                'fontsize',20,'color',[0 0 1],...
                'xoff',0,'yoff',.05);


for i=1:2
```

```
set(figure(i),'Position',[50 50 850 650]);
set(figure(i),'PaperOrientation','portrait','PaperSize',[8.5 7],'
    PaperPositionMode', 'auto', 'PaperType','<custom>');
savestr = ['plot_' num2str(i)];
saveas(figure(i),savestr,'pdf')
end
```