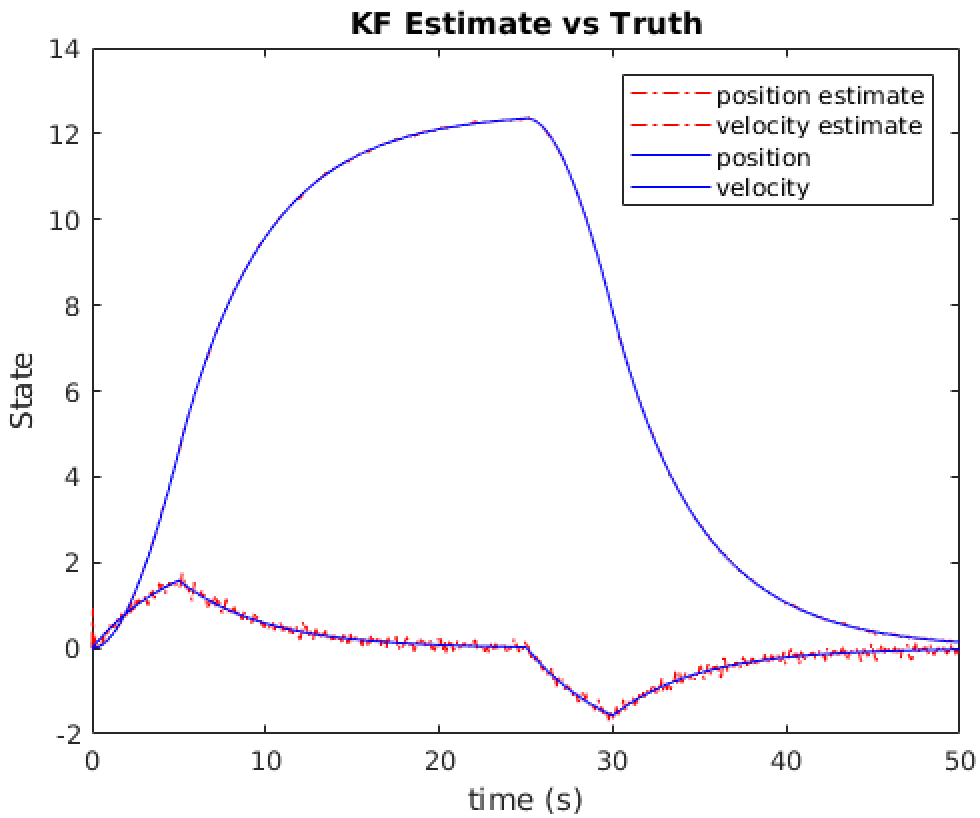
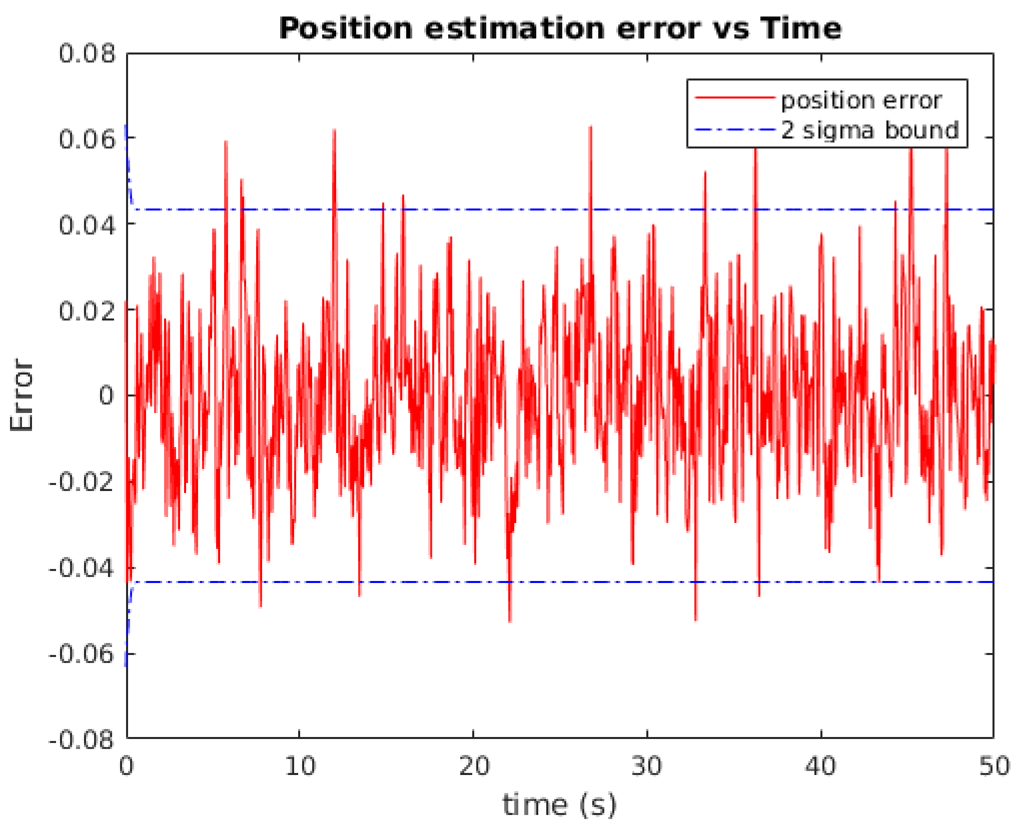
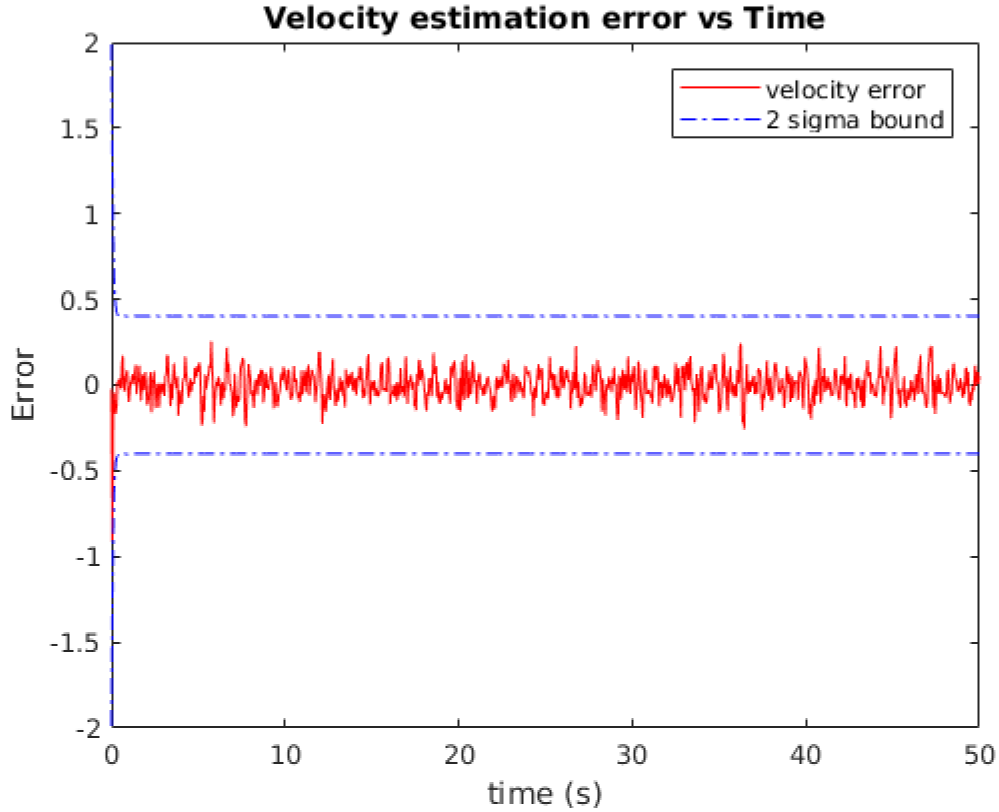
```matlab
% Homework 1 Autonomous Underwater Vehicle (AUV) Kalman Filter
% Jesse Wynn
% September 18, 2017

clc;
clear all;
close all;

% System Parameters

m = 100; % mass (kg)
b = 20; % N-s/m
Ts = 0.05;  % s

% time
t = 0:Ts:50;

% input Force, u(t)
u = zeros(1,length(t));
u(1,1:100) = 50;
u(1,501:600) = -50;

% noise characteristics
Q_t = 0.001;  % measurement noise covariance
% Q_t = 0.01;  % measurement noise covariance
% Q_t = 0.1;  % measurement noise covariance

R_t = [0.0001 0; 0 0.01];   % process noise covariance for pos and vel
% R_t = [0.001 0; 0 0.1];   % process noise covariance for pos and vel
% R_t = [0.01 0; 0 1.0];   % process noise covariance for pos and vel


% Continuous system equations of motion

F = [0 1; 0 -b/m];
G = [0; 1/m];
H = [1 0];
J = [0];

% Create the continuous ss system, sys
sys = ss(F,G,H,J);

% Convert to discrete-time system
sysd = c2d(sys,Ts);

% Obtain difference equation variables
[A B C D] = ssdata(sysd);

% Get the true state X for KF performance comparison
[Y,T,X] = lsim(sysd,u,t,[0; 0]);

% Now do the Kalman Filter stuff...
```

```
% initialize KF variables
% sigma_t = [0.00001 0; 0 0.00001]; % KF covariance
sigma_t = [1 0; 0 1]; % KF covariance
mu_t = [0; 0];

% initialize plot variables
pos_est = zeros(1,length(t));
vel_est = zeros(1,length(t));
K_gain = zeros(2,length(t));
sigma_t_pos = zeros(1,length(t));
sigma_t_vel = zeros(1,length(t));

% Implement the KF Algorithm
for i = 1:length(t)
    % prediction step
    mu_t = A * mu_t + B * u(i);
    sigma_t = A * sigma_t * A' + R_t;

    % update step
    K_t = sigma_t * C' / (C * sigma_t * C' + Q_t(1,1));
    z_t = X(i,1) + randn(1,1) * sqrt(Q_t); % measurement is truth +
 noise
    mu_t = mu_t + K_t * (z_t - C * mu_t);
    sigma_t = (eye(2) - K_t * C) * sigma_t;

    % stuff for plots
    pos_est(1,i) = mu_t(1,1);
    vel_est(1,i) = mu_t(2,1);
    K_gain(1,i) = K_t(1,1);
    K_gain(2,i) = K_t(2,1);
    sigma_t_pos(1,i) = sigma_t(1,1);
    sigma_t_vel(1,i) = sigma_t(2,2);
end

% estimated state mu_est plots
mu_est = [pos_est' vel_est'];

figure(1)
plot(t,mu_est,'-.r',T,X,'b')
title('KF Estimate vs Truth')
xlabel('time (s)')
ylabel('State')
legend('position estimate', 'velocity
 estimate', 'position', 'velocity')

% error plots
pos_error = X(:,1) - mu_est(:,1);
vel_error = X(:,2) - mu_est(:,2);

% generate upper and lower 2 sigma bounds for the error
for i = 1:length(T)
    upper_pos(i,1) = 0 + 2*sqrt(sigma_t_pos(1,i));
    lower_pos(i,1) = 0 - 2*sqrt(sigma_t_pos(1,i));
```

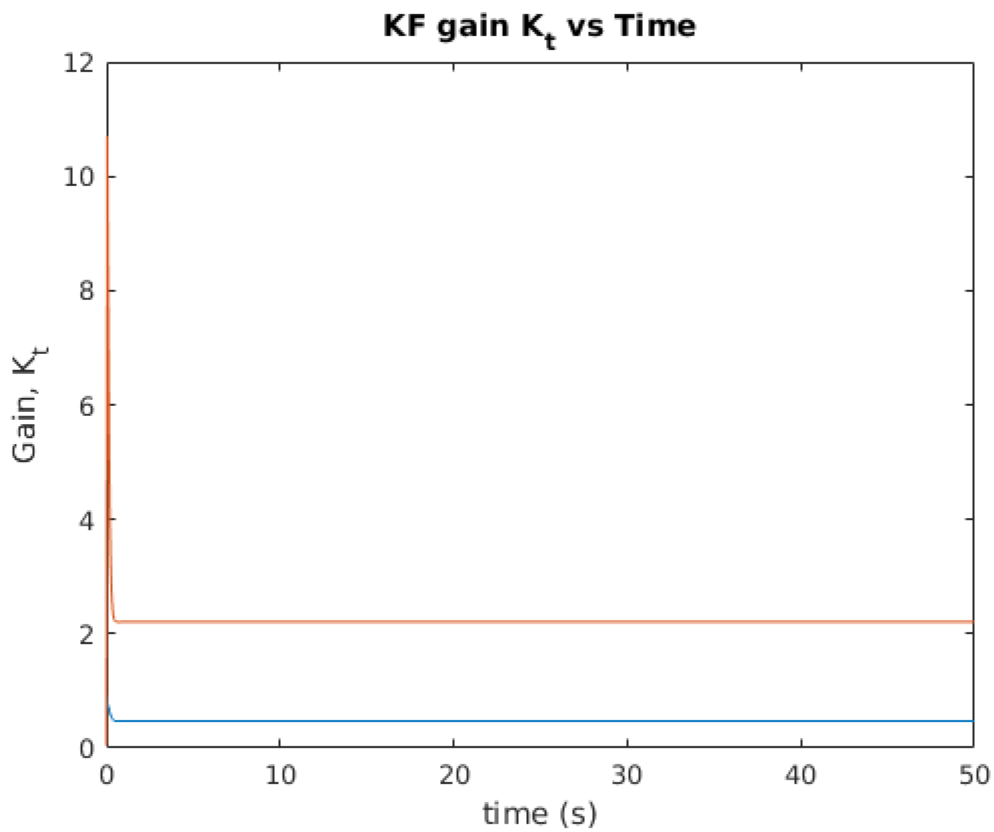
```
    upper_vel(i,1) = 0 + 2*sqrt(sigma_t_vel(1,i));
    lower_vel(i,1) = 0 - 2*sqrt(sigma_t_vel(1,i));
end

% position error plot
figure(2)
plot(T,pos_error,'r',T,upper_pos,'-.b',T,lower_pos,'-.b')
title('Position estimation error vs Time')
xlabel('time (s)')
ylabel('Error')
legend('position error','2 sigma bound')

% velocity error plot
figure(3)
plot(T,vel_error,'r',T,upper_vel,'-.b',T,lower_vel,'-.b')
title('Velocity estimation error vs Time')
xlabel('time (s)')
ylabel('Error')
legend('velocity error','2 sigma bound')

% KF gain plot
figure(4)
plot(t,K_gain)
title('KF gain K_t vs Time')
xlabel('time (s)')
ylabel('Gain, K_t')
```


KF Estimate vs Truth
position estimate
velocity estimate
position
velocity
State
time (s)


Position estimation error vs Time
position error
2 sigma bound
Error
time (s)



Velocity estimation error vs Time
velocity error
2 sigma bound
Error
time (s)



KF gain K_t vs Time
Gain, K_t
time (s)

Published with MATLAB® R2017a