Homework 5 Report

AERO626, Spring 2023

Name: David van Wijk UIN: 932001896

Problem 1 Results

For problem 1, an Extended Kalman Filter (EKF) was implemented in MATLAB using the template provided by Dr. Demars. The code can be found in Appendix A.

Problem 2 Results

The EKF was applied to the estimation of the initial state of an orbiting body under the two-body assumption using angles only measurements. The right-ascension and declination innovations in units of arcseconds are plotted as a function of measurement number in Figure 1. The 3σ for the measurement noise covariance and innovation covariances are also plotted on the same figure. In this plot, the limits were selected such that details throughout the process could be seen, because the 3σ for the innovation covariance is very large at the beginning of the process. Additionally, the innovation covariance and the measurement noise covariances appear to lie on top of each other in the figure - they are very close to the same (especially towards the end of the process) but are distinct nonetheless.

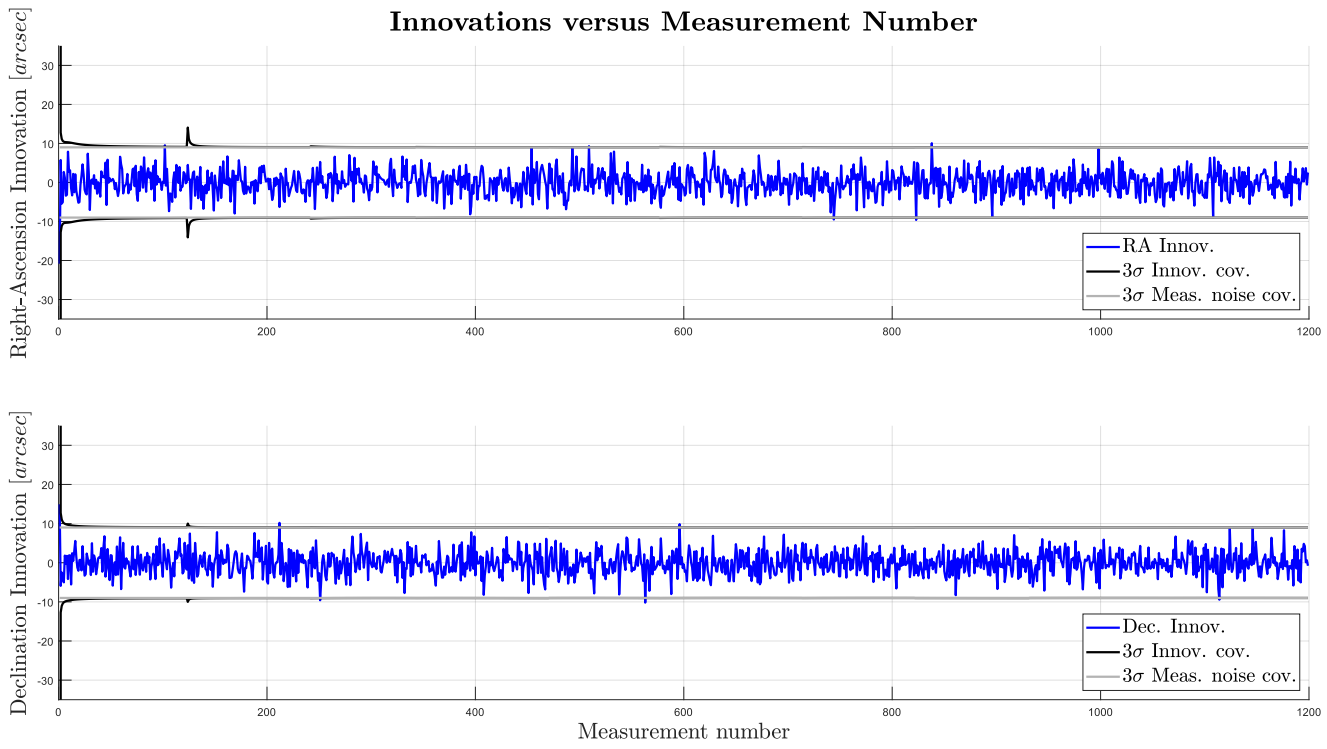

Figure 1: Right-Ascension and Declination versus Measurement Number.

Problem 3 Results

The squared Mahalanobis distance for the innovation as a function of the measurement number is shown in Figure 2. On the plot, the χ_2 values of thresholds for measurement rejection are plotted according to

the probability gates of P_G=95%, P_G=99%, and P_G=99.9%. For a measurement state length of 2, this corresponds to χ_2 values of 5.99, 9.21, and 13.82 respectively.

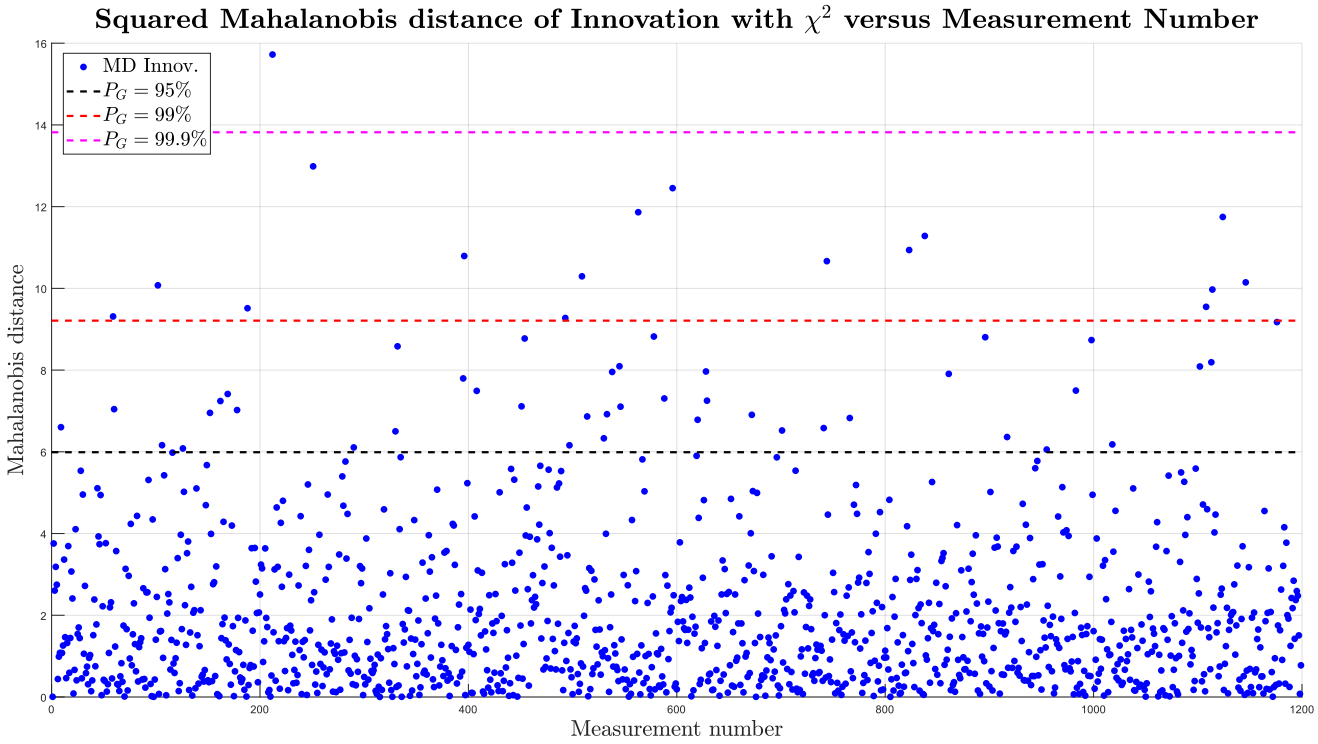

Figure 2: Squared Mahalanobis distance for the innovation versus Measurement Number.

Problem 4 Results

The prior and posterior estimation error for each state (x,y,z position and x,y,z velocity) is plotted on Figure 3 along with the 3σ for prior and posterior standard deviations of the state estimation error covariance. In this plot, the limits were selected such that details throughout the process could be seen, because the 3σ for the estimation error is very large at the beginning of the process.

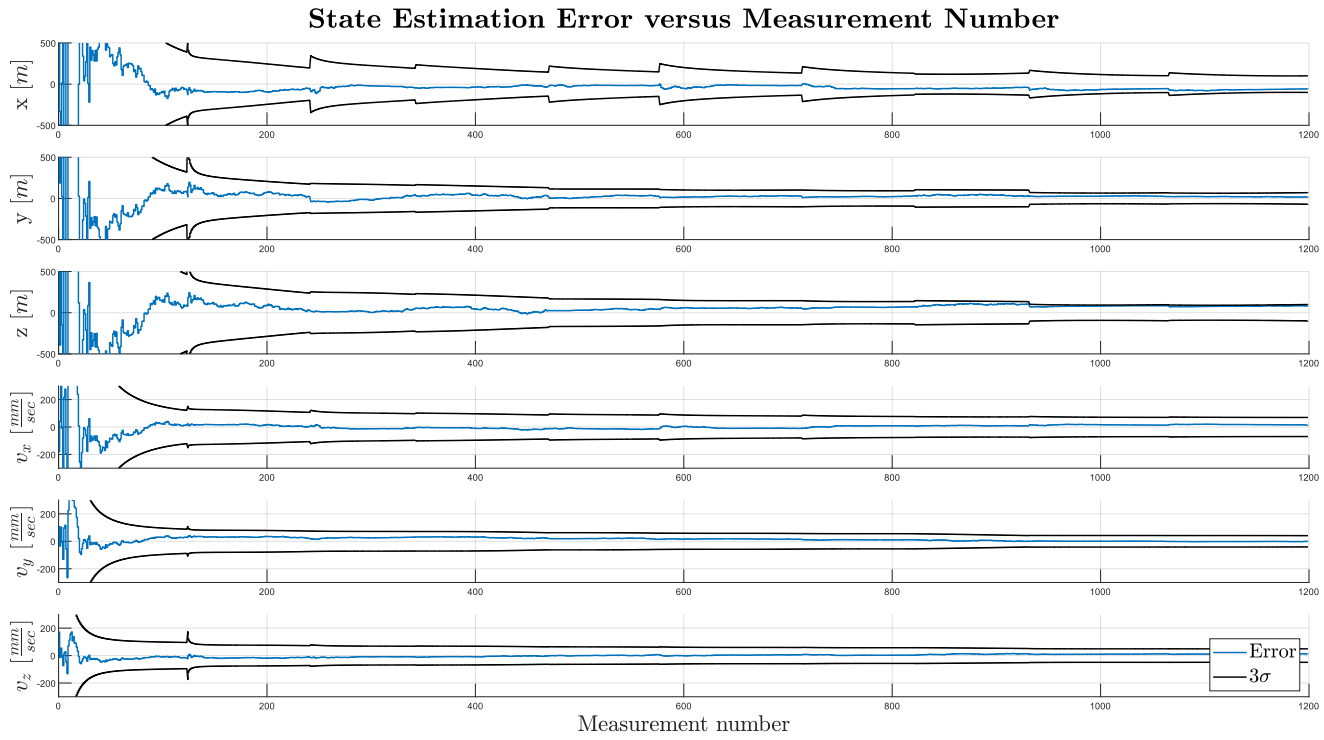

Figure 3: Prior and Posterior Estimation Error for each state versus Measurement Number.

Problem 5 Results

The standard deviations in the states after all the measurements are processed are reported below. This can be confirmed by the output of the MATLAB code in Appendix A.

$$\boldsymbol{\sigma_{xx}} = \begin{bmatrix} 33.5290\ [m] \\ 23.3786\ [m] \\ 33.1328\ [m] \\ 23.1671\ [\frac{mm}{s}] \\ 13.6648\ [\frac{mm}{s}] \\ 16.3893\ [\frac{mm}{s}] \end{bmatrix}$$

Problem 6 Results

The EKF performed very well in estimating the initial state of the spacecraft using angles only data. The final state error after all the measurements were processed was:

$$\boldsymbol{e_{xx}} = \begin{bmatrix} -58.6055\ [m] \\ 19.5065\ [m] \\ 83.9645\ [m] \\ 14.6925\ [\frac{mm}{s}] \\ -0.3828\ [\frac{mm}{s}] \\ 12.0000\ [\frac{mm}{s}] \end{bmatrix}$$

with the standard deviation in the states reported above. We see in Figure 3, that the state error remained within the 3σ intervals for the entire process. Additionally, the innovation covariance approaches the measurement covariance, which is expected. It is interesting to take a closer look at the measurement that exceeds the P_G=99.9% threshold in Figure 2 and see that it corresponds to the declination spike in Figure 1 at measurement number 212. The large deviation in the actual measurement versus the expected

measurement caused this spike, and causes the squared Mahalanobis distance for the innovation to be quite large. However, our filter is able to handle this extraneous measurement quite easily as seen in Figure 3. We also see some interesting behavior when measurements from different nights are processed. There are sharp spikes especially in the x position estimation covariances when a new set of data are processed. It makes sense that these spikes should occur between sets of measurements, and that they be more observable in the position estimates.

A Matlab Code

```
%% AERO 626 Homework 5
%
%   ** BASED ON TEMPLATE PROVIDED BY DR. DEMARS **
%
%   Texas A&M University
%   Aerospace Engineering
%   van Wijk, David

%% Kalman Filter Implementation

clear; close all; clc

plot_flag = true;
opts = odeset('AbsTol',1e-9,'RelTol',1e-9);

% NOTATION:
%   tkm1   = t_{k-1}         time at the (k-1)th time
%   mxkm1  = m_{x,k-1}^{+}   posterior mean at the (k-1)th time
%   Pxxkm1 = P_{xx,k-1}^{+}  posterior covariance at the (k-1)th time
%   tk     = t_{k}           time at the kth time
%   mxkm   = m_{x,k}^{-}     prior mean at the kth time
%   Pxxkm  = P_{xx,k}^{-}    prior covariance at the kth time
%   mxkp   = m_{x,k}^{+}     posterior mean at the kth time
%   Pxxkp  = P_{xx,k}^{+}    posterior covariance at the kth time
%   Pzzkm  = P_{zz,k}^{-}    innovation covariance at the kth time
%   Pvvk   = P_{vv,k}        measurement noise covariance at the kth time

% constants
GM = 3.986004415e5;          % [km^3/s^2]

% conversion factors
asc2deg = 1.0/3600.0;        % [arcsec] -> [deg]
deg2rad = pi/180.0;          % [deg] -> [rad]
asc2rad = asc2deg*deg2rad;   % [arcsec] -> [rad]
rad2asc = 3600.0*180.0/pi;   % [rad] -> [arcsec]
rad2deg = 180.0/pi;          % [rad] -> [deg]

% load data file
load('data_HW05','T','Z','Pvv','Robsv','Xtrue')
% DATA PROVIDED ARE:
%   T     = (m x 1) array of observation times [sec]
%   Z     = (2 x m) array of right-ascension and declination observations
    [arcsec]
%   Pvv   = (2 x 2 x m) array of measurement noise covariances [arcsec^2]
%   Robsv = (3 x m) array of observer position in inertial frame [km]
%   Xtrue = (6 x m) array of true object position and velocity [km] and [
   km/s]
% m = 1199 for this dataset
% there are 123, 118, 101, 128, 106, 137, 108, 110, 134, 134 measurements
    on
% each of the ten consecutive nights, respectively
```

```
% given initial conditions
t0   = 0.0;
mx0  = [-24864.5000; -9288.5000; 16.5000; 0.7726; -2.0677; -3.1867];
Pxx0 = [(.2)^2*eye(3), zeros(3);
       zeros(3), (2e-3)^2*eye(3)];

% declare storage space for saving state estimation error information
xcount  = 0;
xstore  = nan(6,2*length(T)-1);
txstore = nan(1,2*length(T)-1);
kxstore = nan(1,2*length(T)-1);
mxstore = nan(6,2*length(T)-1);
exstore = nan(6,2*length(T)-1);
sxstore = nan(6,2*length(T)-1);

% declare storage space
zcount  = 0;
zstore  = nan(2,length(T));
tzstore = nan(1,length(T));
kzstore = nan(1,length(T));
mzstore = nan(2,length(T));
ezstore = nan(2,length(T));
dzstore = nan(1,length(T));
szstore = nan(2,length(T));
svstore = nan(2,length(T));

% store initial data
xcount             = xcount + 1;
txstore(:,xcount) = T(1);
kxstore(:,xcount) = 0;
mxstore(:,xcount) = mx0;
sxstore(:,xcount) = sqrt(diag(Pxx0));

% measurement noise
Hv = eye(2);

% initialize time, mean, and covariance for the EKF
tkm1   = t0;
mxkm1  = mx0;
Pxxkm1 = Pxx0;

% loop over the number of data points
for k = 1:length(T)
    tk   = T(k);       % time of the current measurement to process [sec]
    zk   = Z(:,k);     % current measurement to process [arcsec]
    Pvvk = Pvv(:,:,k); % measurement noise covariance [arcsec^2]
    rk   = Robsv(:,k); % inertial position of the observer [km]

    % unpack the truth -- this cannot be used in the filter, only for
       analysis
    xk = Xtrue(:,k);   % true state of the object [km and km/s]
```

```
% propagate the mean and covariance
x_full = [mxkm1; reshape(Pxxkm1,36,1)];
[~,X] = ode45(@(t,x) TwoBodyProp(t,x,GM),[tkm1,tk],x_full,opts);

mxkm = X(end,1:6)';
Pxxkm = reshape(X(end,7:42),6,6);

% store a priori state information for analysis
xcount            = xcount + 1;
xstore(:,xcount)  = xk;
txstore(:,xcount) = tk;
kxstore(:,xcount) = k;
mxstore(:,xcount) = mxkm;
exstore(:,xcount) = xk - mxkm;
sxstore(:,xcount) = sqrt(diag(Pxxkm));

% compute the estimated measurement
%   form the relative position, then compute the expected measurement
%   change the units of the reference measurement to [arcsec]

relPos = mxkm(1:3) - Robsv(:,k);
x = relPos(1);
y = relPos(2);
z = relPos(3);
h = [(atan2(y,x)); (atan2(z,(sqrt(x^2+y^2))))]; % [rad]
mzkm = h*rad2asc;

% compute the measurement Jacobian
%   change the units of the Jacobian to [arcsec]

Hxk = computeHx(relPos);
Hxk = Hxk*rad2asc;

% update the mean and covariance

Pxzkm = Pxxkm*Hxk';
Pzzkm = Hxk*Pxxkm*Hxk' + Hv*Pvvk*Hv';
Kk = Pxzkm/Pzzkm;
mxkp = mxkm + Kk*(zk - mzkm);
Pxxkp = Pxxkm - Pxzkm*Kk' - Kk*(Pxzkm)' + Kk*(Pzzkm)*Kk';

% store a posteriori state information for analysis
xcount            = xcount + 1;
xstore(:,xcount)  = Xtrue(:,k);
txstore(:,xcount) = tk;
kxstore(:,xcount) = k;
mxstore(:,xcount) = mxkp;
exstore(:,xcount) = Xtrue(:,k) - mxkp;
sxstore(:,xcount) = sqrt(diag(Pxxkp));

% store measurement information for analysis
zcount            = zcount + 1;
zstore(:,zcount)  = zk;
```

```
    tzstore(:,zcount) = tk;
    kzstore(:,zcount) = k;
    mzstore(:,zcount) = mzkm;
    ezstore(:,zcount) = zk - mzkm;
    dzstore(:,zcount) = (zk - mzkm)'*(Pzzkm\(zk - mzkm));
    szstore(:,zcount) = sqrt(diag(Pzzkm));
    svstore(:,zcount) = sqrt(diag(Pvvk));

    % cycle the time, mean, and covariance for the next step of the EKF
    tkm1   = tk;
    mxkm1  = mxkp;
    Pxxkm1 = Pxxkp;
end

finalSTD_state = sqrt(diag(Pxxkm1));
disp('Final standard deviations of the posterior covariance in the
   position [m]:')
finalSTD_state(1:3)*1e3
disp('Final standard deviations of the posterior covariance in the
   velocity [mm/s]:')
finalSTD_state(4:6)*1e6

%% Plots

if plot_flag

    measx = 1:length(T);
    xaxis_sz = 20; yaxis_sz = 20; legend_sz = 18;
    std_plot = 3; txt = [num2str(std_plot) '$\sigma$'];

    % Plots for Part 2
    ylim_RA  = 35;
    ylim_DEC = 35;

    figure; grid on; set(gcf, 'WindowState', 'maximized');
    subplot(2,1,1); hold on; grid on; ylim([-ylim_RA ylim_RA])
    title('\textbf{Innovations versus Measurement Number}','Fontsize',25,
       'interpreter','latex')
    a1 = plot(measx,ezstore(1,:),'-','Color','b','LineWidth',2,'
       MarkerSize',20);
    b1 = plot(measx,std_plot*szstore(1,:),'-','Color','k','LineWidth',2,'
       MarkerSize',20);
    b2 = plot(measx,std_plot*svstore(1,:),'-','Color',[.7 .7 .7],'
       LineWidth',2,'MarkerSize',20);
    plot(measx,-std_plot*szstore(1,:),'-','Color','k','LineWidth',2,'
       MarkerSize',20);
    plot(measx,-std_plot*svstore(1,:),'-','Color',[.7 .7 .7],'LineWidth'
       ,2,'MarkerSize',20);
    ylabel('Right-Ascension Innovation [$arcsec$]','Fontsize',yaxis_sz,'
       interpreter','latex')
    legendtxt = {'RA Innov.', [txt  ' Innov. cov.'],[txt ' Meas. noise
       cov.']};
    legend([a1 b1 b2],legendtxt,'FontSize',legend_sz,'interpreter','latex
```

```
    ','location','southeast')

subplot(2,1,2); hold on; grid on; ylim([-ylim_DEC ylim_DEC])
a2 = plot(measx,ezstore(2,:),'-','Color','b','LineWidth',2,'
    MarkerSize',20);
b3 = plot(measx,std_plot*szstore(2,:),'-','Color','k','LineWidth',2,'
    MarkerSize',20);
b4 = plot(measx,std_plot*svstore(2,:),'-','Color',[.7 .7 .7],'
    LineWidth',2,'MarkerSize',20);
plot(measx,-std_plot*szstore(2,:),'-','Color','k','LineWidth',2,'
    MarkerSize',20);
plot(measx,-std_plot*svstore(2,:),'-','Color',[.7 .7 .7],'LineWidth'
    ,2,'MarkerSize',20);
ylabel('Declination Innovation [$arcsec$]','Fontsize',yaxis_sz,'
    interpreter','latex')
legendtxt = {'Dec. Innov.', [txt  ' Innov. cov.'],[txt ' Meas. noise
    cov.']};
legend([a2 b3 b4],legendtxt,'FontSize',legend_sz,'interpreter','latex
    ','location','southeast')
xlabel('Measurement number','Fontsize',xaxis_sz,'interpreter','latex'
    )

% Plots for Part 3

figure; grid on; set(gcf, 'WindowState', 'maximized'); hold on;
title('\textbf{Squared Mahalanobis distance of Innovation with $\chi
    ^{2}$ versus Measurement Number}','Fontsize',25,'interpreter','
    latex')
b1 = scatter(measx,dzstore,'filled','MarkerFaceColor','b');
a1 = line([0,measx(end)],[5.99 5.99],'LineStyle','--','Color','k','
    LineWidth',2);
a2 = line([0,measx(end)],[9.21 9.21],'LineStyle','--','Color','r','
    LineWidth',2);
a3 = line([0,measx(end)],[13.82 13.82],'LineStyle','--','Color','m','
    LineWidth',2);
ylabel('Mahalanobis distance','Fontsize',yaxis_sz,'interpreter','
    latex')
xlabel('Measurement number','Fontsize',xaxis_sz,'interpreter','latex'
    )
legendtxt = {'MD Innov.', '$P_{G} = 95\%$', '$P_{G} = 99\%$', '$P_{G}
     = 99.9\%$'};
legend([b1 a1 a2 a3],legendtxt,'FontSize',legend_sz,'interpreter','
    latex','location','northwest')

% Plots for Part 4
err_line_opts = {'-','LineWidth',1.3};
std_line_opts = {'-','LineWidth',1.3,'Color','k'};

ylim_pos = 500;
ylim_vel = 300;

exstore(1:3,:) = exstore(1:3,:) * 1e3;  % convert from km to m
exstore(4:6,:) = exstore(4:6,:) * 1e6;  % convert from km/s to mm/s
```

```
sxstore(1:3,:) = sxstore(1:3,:) * 1e3;  % convert from km to m
sxstore(4:6,:) = sxstore(4:6,:) * 1e6;  % convert from km/s to mm/s

opts_pts1 = {80,'b',"o"};
opts_pts2 = {80,'r',"o"};

measx1 = sort([measx measx]);

figure; grid on; set(gcf, 'WindowState', 'maximized');
subplot(6,1,1); hold on; grid on; ylim([-ylim_pos ylim_pos])
title('\textbf{State Estimation Error versus Measurement Number}','
   Fontsize',25,'interpreter','latex')
plot(measx1,exstore(1,2:end),err_line_opts{:})
plot(measx1,std_plot*sxstore(1,2:end),std_line_opts{:})
plot(measx1,-std_plot*sxstore(1,2:end),std_line_opts{:})
ylabel('x [$m$]','Fontsize',yaxis_sz,'interpreter','latex')

subplot(6,1,2); hold on; grid on; ylim([-ylim_pos ylim_pos])
plot(measx1,exstore(2,2:end),err_line_opts{:})
plot(measx1,std_plot*sxstore(2,2:end),std_line_opts{:})
plot(measx1,-std_plot*sxstore(2,2:end),std_line_opts{:})
ylabel('y [$m$]','Fontsize',yaxis_sz,'interpreter','latex')

subplot(6,1,3); hold on; grid on; ylim([-ylim_pos ylim_pos])
plot(measx1,exstore(3,2:end),err_line_opts{:})
plot(measx1,std_plot*sxstore(3,2:end),std_line_opts{:})
plot(measx1,-std_plot*sxstore(3,2:end),std_line_opts{:})
ylabel('z [$m$]','Fontsize',yaxis_sz,'interpreter','latex')

subplot(6,1,4); hold on; grid on; ylim([-ylim_vel ylim_vel])
plot(measx1,exstore(4,2:end),err_line_opts{:})
plot(measx1,std_plot*sxstore(4,2:end),std_line_opts{:})
plot(measx1,-std_plot*sxstore(4,2:end),std_line_opts{:})
ylabel('$v_x$ [$\frac{mm}{sec}$]','Fontsize',yaxis_sz,'interpreter','
   latex')

subplot(6,1,5); hold on; grid on; ylim([-ylim_vel ylim_vel])
plot(measx1,exstore(5,2:end),err_line_opts{:})
plot(measx1,std_plot*sxstore(5,2:end),std_line_opts{:})
plot(measx1,-std_plot*sxstore(5,2:end),std_line_opts{:})
ylabel('$v_y$ [$\frac{mm}{sec}$]','Fontsize',yaxis_sz,'interpreter','
   latex')

subplot(6,1,6); hold on; grid on; ylim([-ylim_vel ylim_vel])
a1 = plot(measx1,exstore(6,2:end),err_line_opts{:});
a2 = plot(measx1,std_plot*sxstore(6,2:end),std_line_opts{:});
plot(measx1,-std_plot*sxstore(6,2:end),std_line_opts{:})
ylabel('$v_z$ [$\frac{mm}{sec}$]','Fontsize',yaxis_sz,'interpreter','
   latex')
xlabel('Measurement number','Fontsize',xaxis_sz,'interpreter','latex'
   )
legendtxt = {'Error',txt};
```

```
    legend([a1 a2],legendtxt,'FontSize',legend_sz,'interpreter','latex','
       location','southeast')

end


%% Functions

function [Hx] = computeHx(x_vect)
x = x_vect(1);
y = x_vect(2);
z = x_vect(3);
a = x^2+y^2;
b = x^2+y^2+z^2;

Hx = [-y/a x/a 0 0 0 0;
    -(x*z)/(b*sqrt(a)) -(y*z)/(b*sqrt(a)) sqrt(a)/b  0 0 0];
end

function dx = TwoBodyProp(~,x,mu)
% Propagate the dynamics
r = norm(x(1:3));
dx(1:3,1) = x(4:6);
dx(4:6,1) = -(mu/r^3)*x(1:3);

% Construct Fx
A = mu*((3*(x(1:3)*x(1:3)'))/r^5 - (eye(3)/r^3));
Fx = [zeros(3,3) eye(3); A zeros(3,3)];

% Extract Pxx & reshape
Pxx = x(7:end);
Pxx = reshape(Pxx,6,6);

% Propagate Pxx
result = Fx*Pxx + Pxx*Fx';

% Reshape back to column
result = reshape(result,36,1);
dx(7:42,1) = result;

end
```