Homework 3 Report

AERO626, Spring 2023

Name: David van Wijk UIN: 932001896

Problem 1 Results

The spring-mass problem in Section 2.7.1 of the course notes was coded up in MATLAB. The code can be found in Appendix A. The reference state after the fourth iteration was:

$$\mathbf{x}_0^* = \begin{bmatrix} 3.0002 \\ 0.0012 \end{bmatrix}$$

Figures Figure 1 and Figure 2 show the range and range-rate residuals at each iteration of the process.

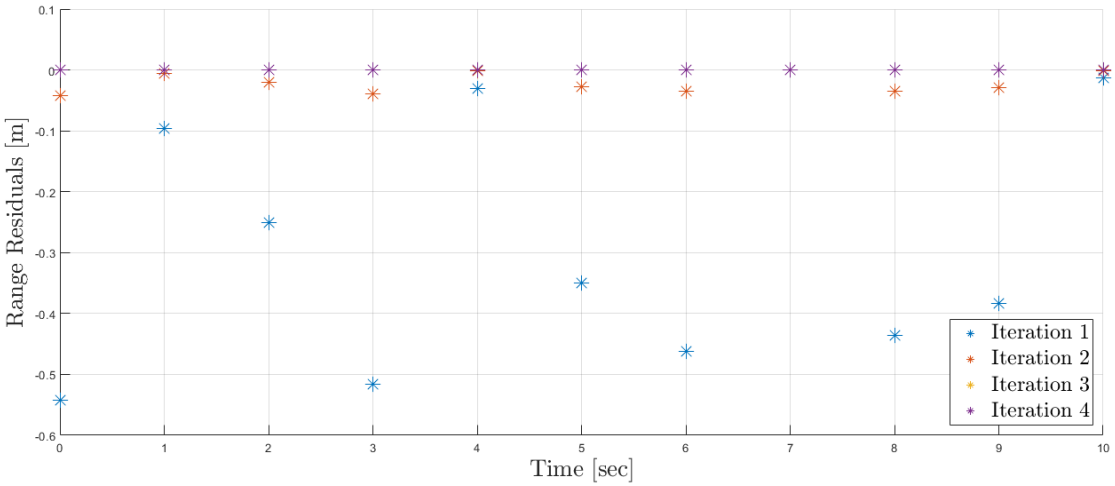

Figure 1: Range Residuals.

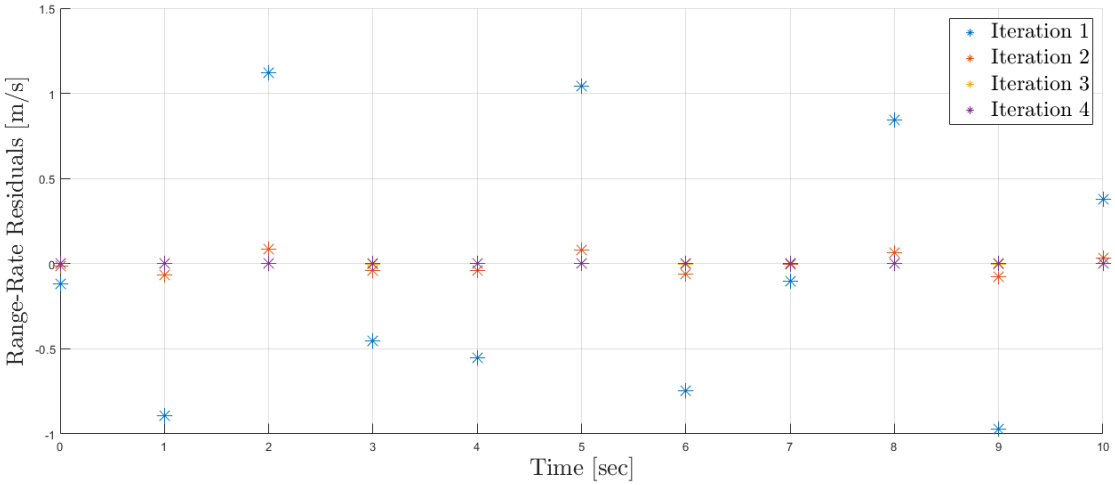

Figure 2: Range-Rate Residuals.

Problem 5 Results

The derivations of the necessary equations for this problem (i.e. solutions to Problems 2-4) can be found below the analysis of Problem 5.

Part A

The same batch processor was applied to the estimation of the initial state of an orbiting body under the two-body assumption using angles only measurements. Again, this was implemented in MATLAB (Appendix A). The right ascension and declination residuals at each iteration are plotted in Figure 3 and Figure 4. The reference state after the fifth iteration is:

$$\mathbf{x}_0^* = \begin{bmatrix} 31689 \\ 23861 \\ 1939.1 \\ -1.7911 \\ 2.5124 \\ 0.2268 \end{bmatrix}$$

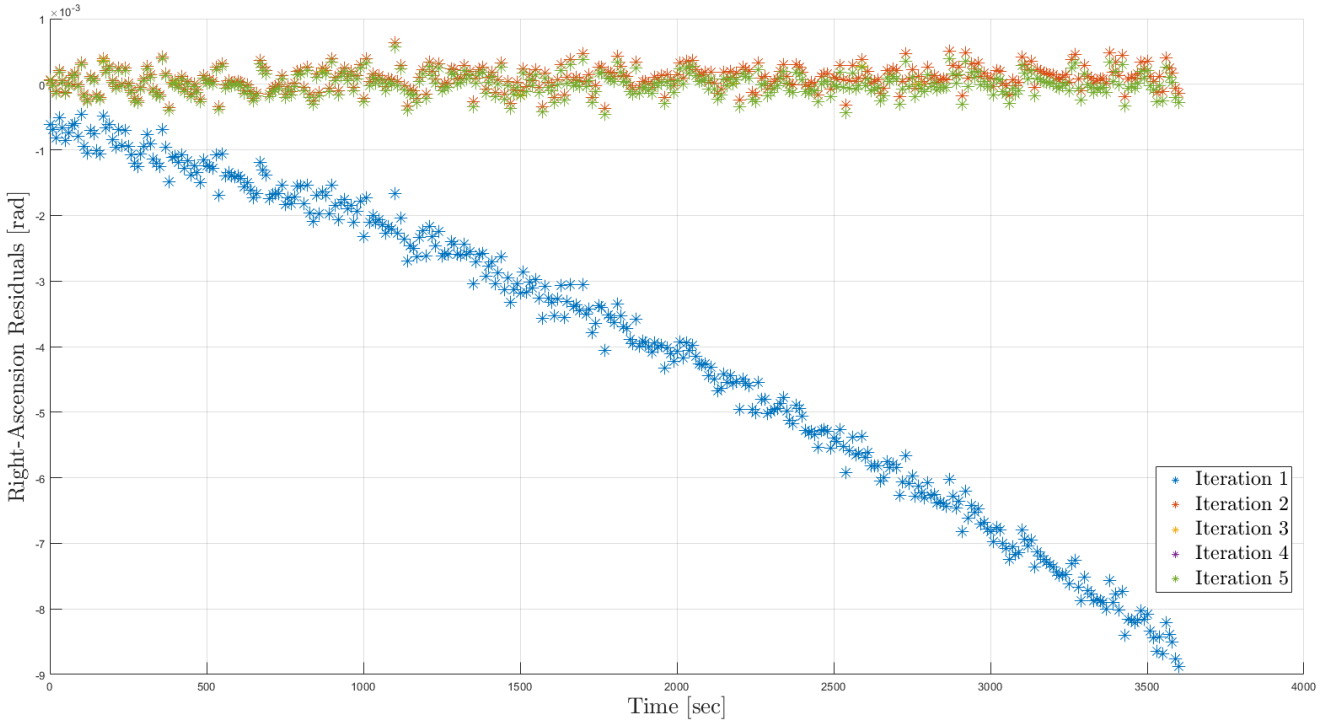

Figure 3: Right-Ascension Residuals per Iteration.

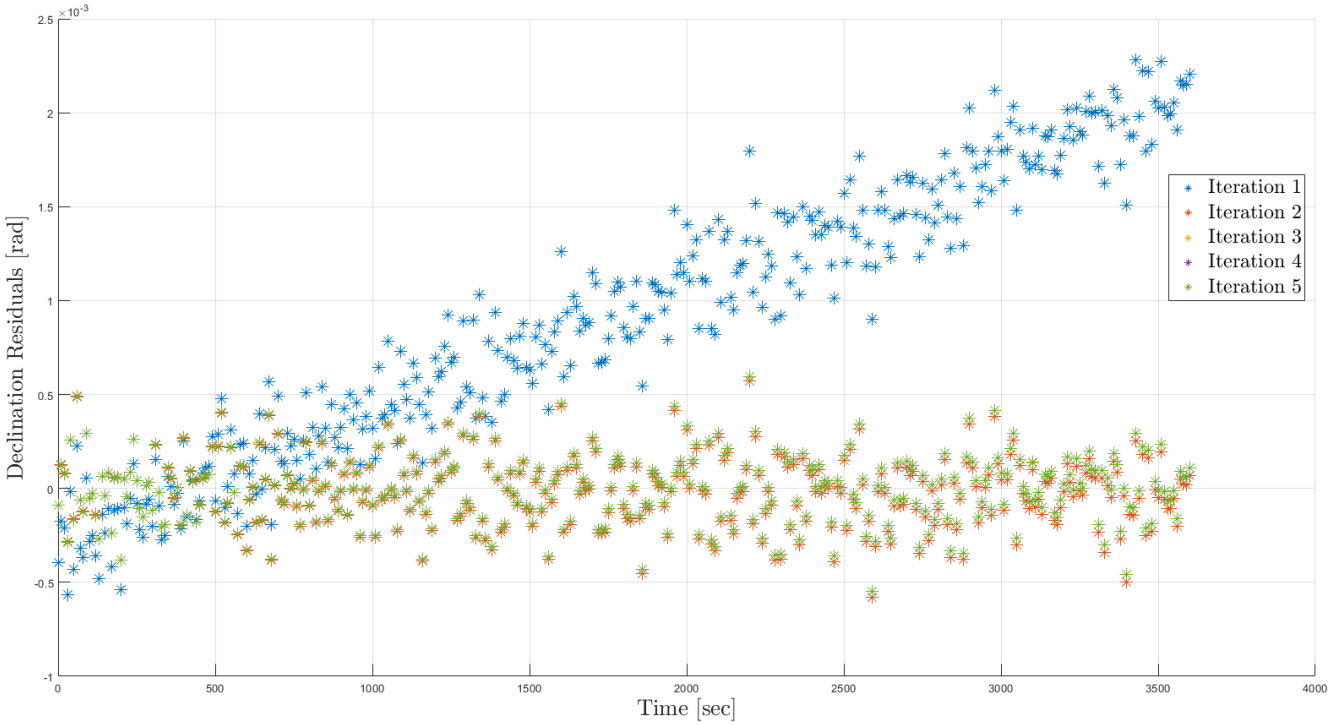

Figure 4: Declination Residuals per Iteration.

Using the plots we can say it is likely that the process has converged because the residuals and the error reach a steady-state. Additional iterations do not improve the estimate of the initial state. This is not a rigorous evaluation of convergence however. We use numerical analysis to determine if the process has converged. To evaluate convergence of the filter, the root mean square (RMS) of the error is computed at each iteration and reported in a tabular form, Table 1. Here the average RMS is reported rather than reporting the two-dimensional RMS for declination and right-ascension. We see that the RMS converges to a value that is very close to the measurement uncertainty, obtained by taking the square root of the diagonal elements of the measurement noise covariance. In this case $P_{vv,l}$ is constant throughout the process.

Table 1: Root Mean Square of Error per Iteration.

Iteration	**1**	**2**	**3**	**4**	**5**
RMS (avg.)	.0029	1.8484e-04	1.7201e-04	1.7201e-04	1.7201e-04

The error with respect to the true state of the object per iteration is plotted in Figure 5. For the error, we subtract off the true state from the estimated initial state. We can see that the error in each state flattens out to a constant value. Additionally, the Mahalanobis distance at each iteration is plotted in Figure 6. This is a scalar measure of how many multi-dimensional standard deviations away from the true state that our estimate is. For this case, since the state vector has a length of 6, a Mahalanobis distance of 6 is a good rule of thumb for adequate performance of the estimator. Therefore we can see that our estimator does a fairly decent job with the data it is given as it settles on a Mahalanobis distance of around 3.

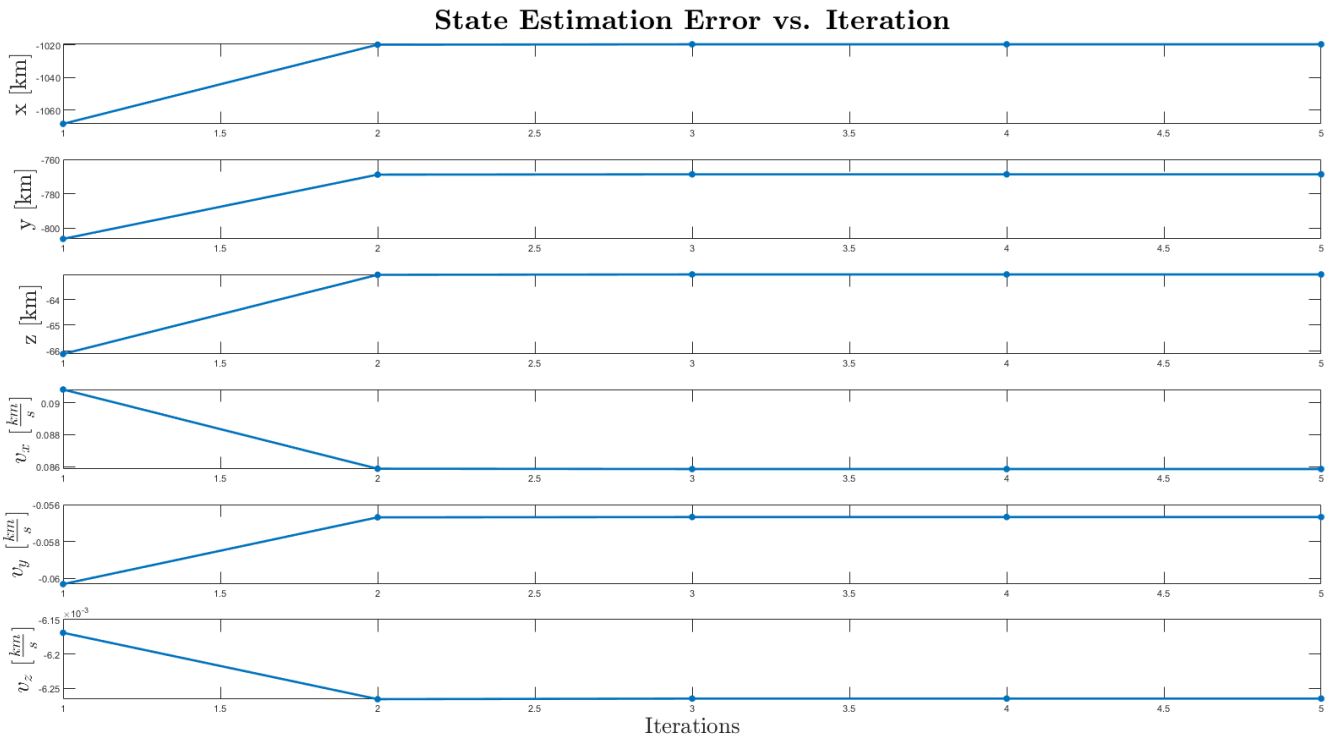

Figure 5: State estimation error per Iteration.

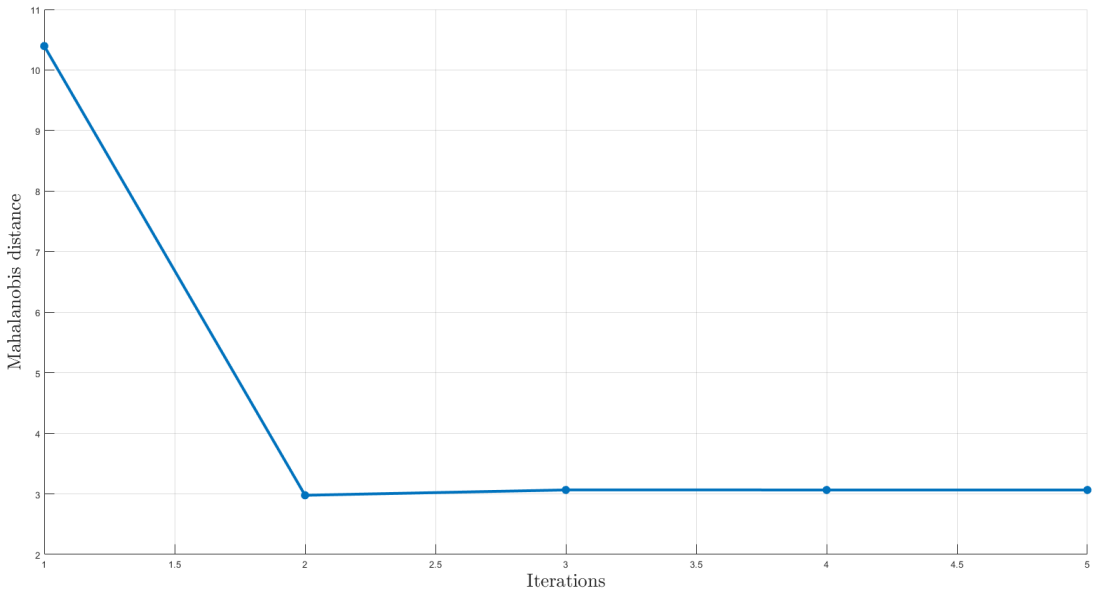

Figure 6: Mahalanobis distance per Iteration.

Part B

The same analysis was repeated for an initial reference state that was further away from the true value of the initial state. In this case, it takes the filter more iterations to converge, which makes intuitive sense, but it does converge nonetheless, with the reference state after the fifth iteration being:

$$\mathbf{x}_0^* = \begin{bmatrix} 31689 \\ 23861 \\ 1939.1 \\ -1.7911 \\ 2.5124 \\ 0.2268 \end{bmatrix}$$

We can see that here the residuals stray far from zero in the first two iterations, but still improve over the number of iterations. Likewise, the state error is larger at the beginning but eventually converges to the exact same values as in Part A, which makes sense since the estimate that we obtain using this worse initial guess converges to the same value as in Part A as well. The Mahalanobis distance starts off very large, but eventually converges to the same scalar value of around 3 as in the previous part. Again, the RMS of the error is tabularized (Table 2), showing that after the 5th iteration we obtain an RMS that is very close to the measurement uncertainty.

Table 2: Root Mean Square of Error per Iteration.

Iteration	**1**	**2**	**3**	**4**	**5**
RMS (avg.)	0.0365	0.0144	0.0017	2.0550e-04	1.7201e-04

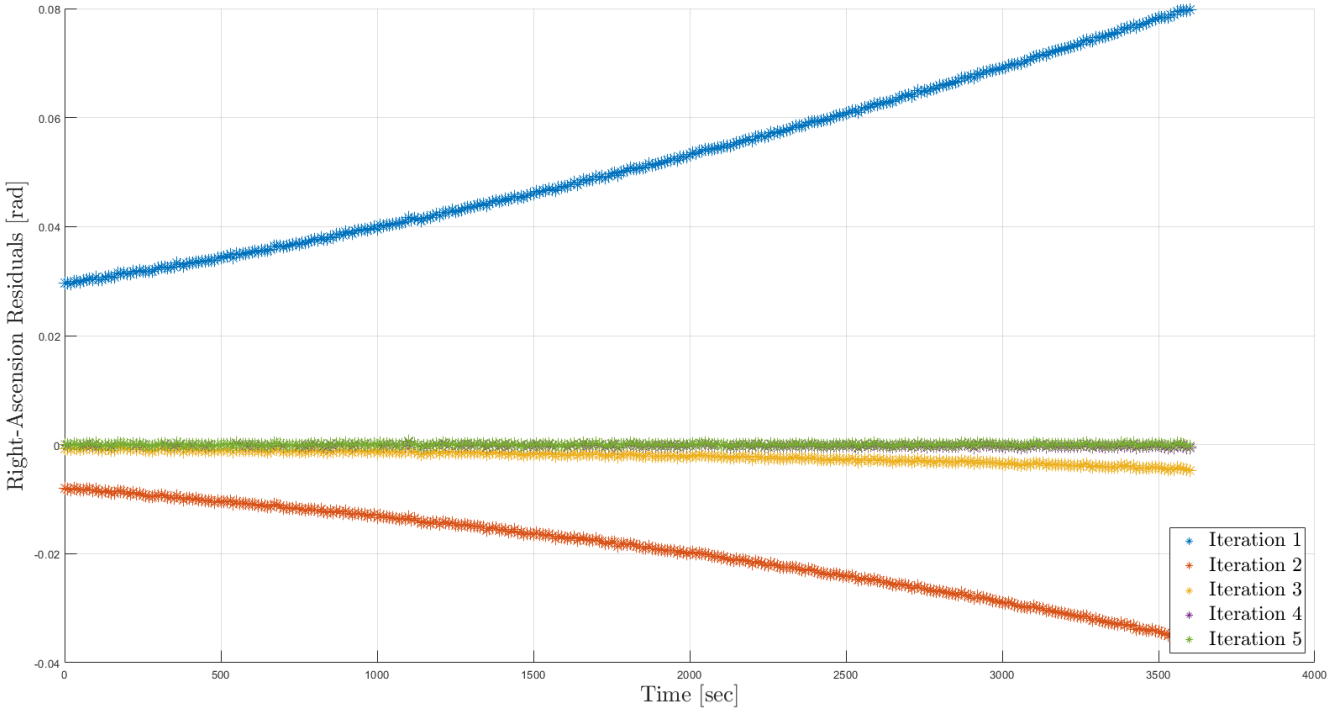

Figure 7: Right-Ascension Residuals per Iteration.

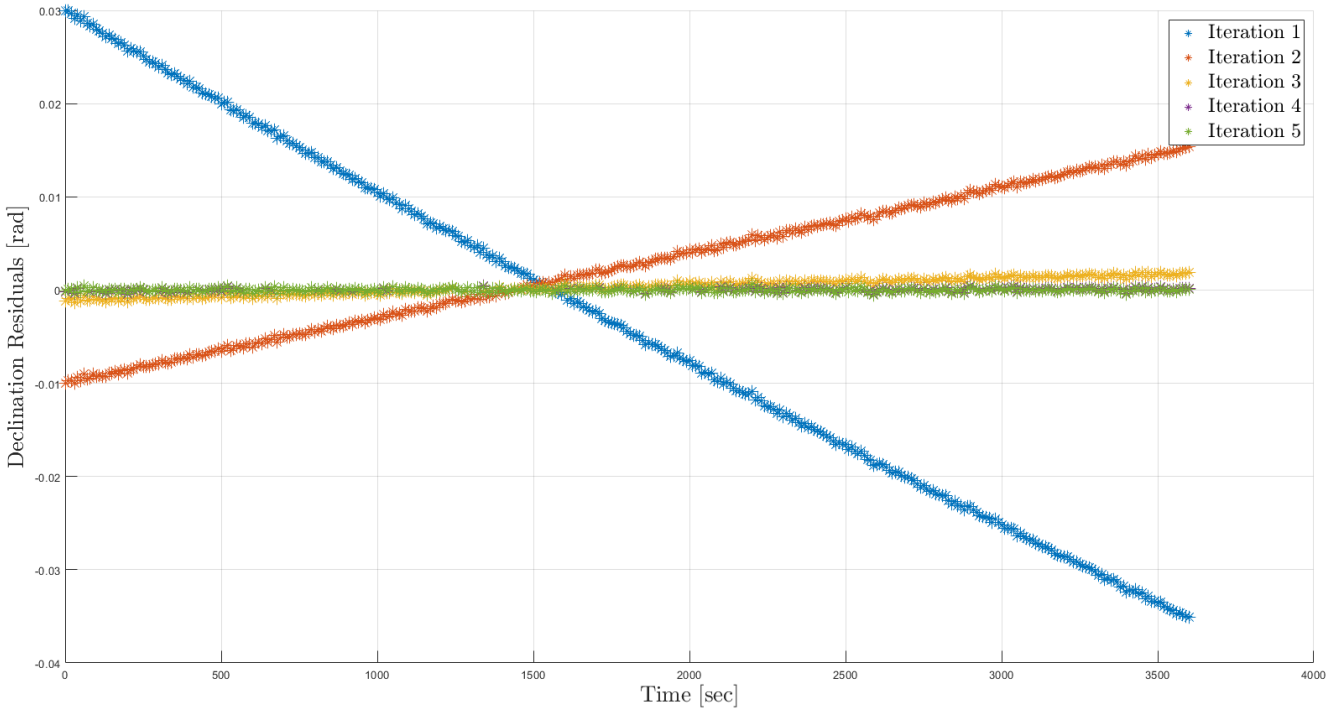

Figure 8: Declination Residuals per Iteration.

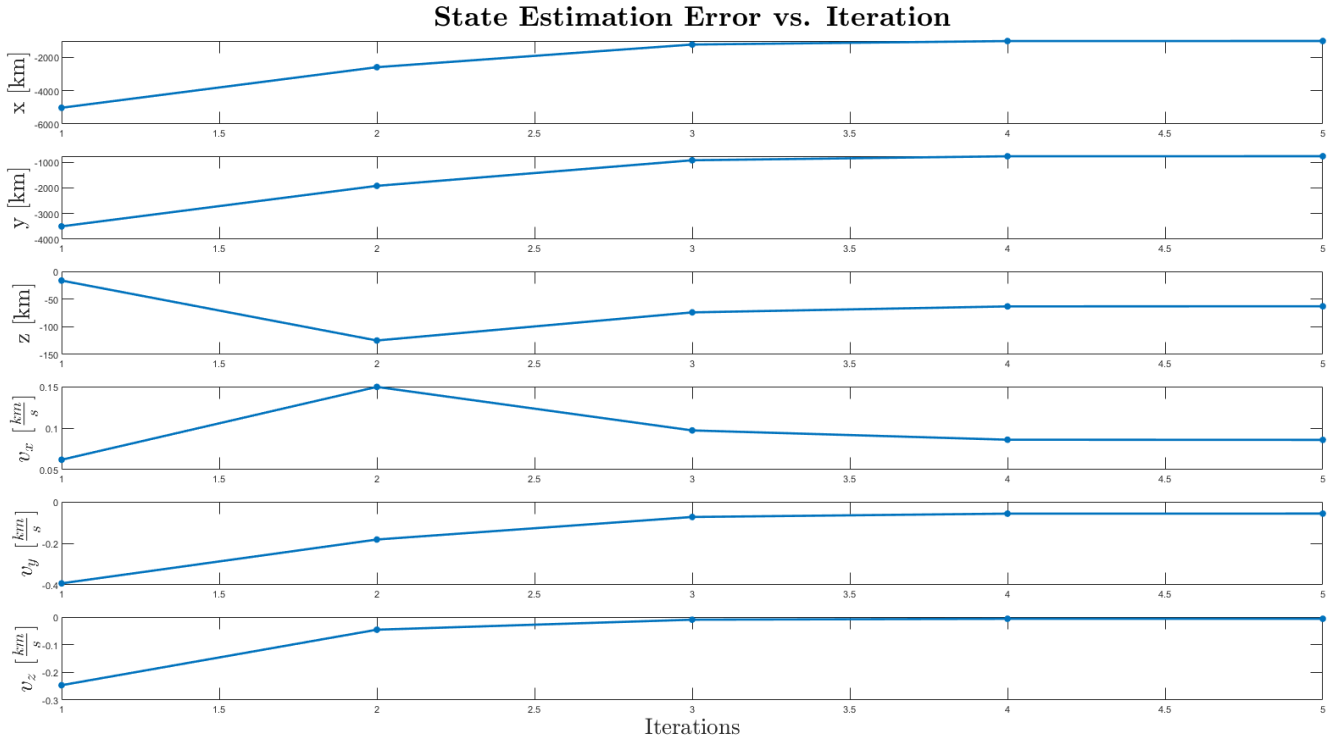

Figure 9: State estimation error per Iteration.

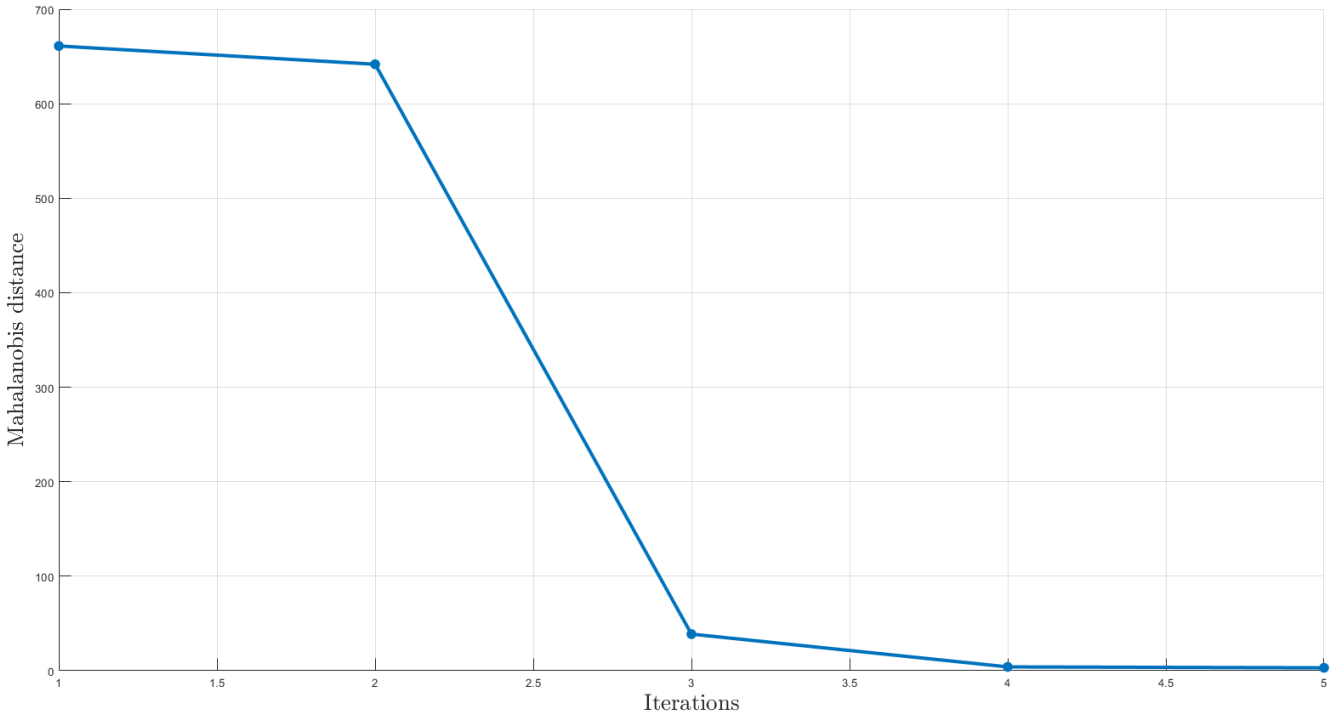

Figure 10: Mahalanobis distance per Iteration.

A Matlab Code

```
%% AERO 626 Homework #3
% Spring 2023
% David van Wijk

%% Problem 1: Spring-Mass Cart
clc;

% Plotting
xaxis_sz = 20; yaxis_sz = 20; legend_sz = 18;
plot_flag = false;

% Parameters
m = 1.5;  %[kg]
k1 = 2.5; k2 = 3.7; %[N/m]
h = 5.4;  x0 = 3.0; %[m]
v0 = 0.0; %[m/s]

x_star = [4; .2];
del_x0 = [0; 0];
Pxx0_bar = [1000 0; 0 100];
data = load('data_HW03_p1.mat');

omega = sqrt((k1 + k2)/m);
F = [0 1; -omega^2 0];
Pvv = eye(2);

n = 4;
for i = 1:n
    CapLambda = (Pxx0_bar^-1);
    Lambda = CapLambda*del_x0;
    t0 = 0;
    h_array = zeros(length(data.hw3_p1_data),2);
    for j = 1:length(data.hw3_p1_data)
        t_l = data.hw3_p1_data(j,1);
        z_l = data.hw3_p1_data(j,2:3)';
        Pvv_l = Pvv;

        Phi = [cos(omega*(t_l-t0)) (1/omega)*(sin(omega*(t_l-t0)));
            -omega*(sin(omega*(t_l-t0))) cos(omega*(t_l-t0))];

        x_l_star = Phi*x_star;

        x = x_l_star(1);
        v = x_l_star(2);
        denom = sqrt(x^2 + h^2);
        H = [x/denom, 0; v/denom - (x^2*v)/(denom^3), x/denom];
        A = H*Phi;
        B = z_l - [denom ; (x*v)/denom];
        Lambda = Lambda + (A)'*(Pvv_l^-1)*(B);
        CapLambda = CapLambda + (A)'*(Pvv_l^-1)*(A);
        h_array(j,:) = [denom (x*v)/denom];
```

```
    end
    Pxx0 = CapLambda^-1;
    del_x0_hat = CapLambda\Lambda;
    x_star = x_star + del_x0_hat;
    del_x0 = del_x0 - del_x0_hat;
    disp(['The reference state after ', num2str(i), ' iterations is:'])
    x_star
    % Plot residuals

    del_z = data.hw3_p1_data(:,2:3) - h_array;
    legend_txt = ['Iteration ',num2str(i)];

    if plot_flag
        figure(1); hold on; grid on; set(gcf, 'WindowState', 'maximized')
           ;
        xlabel('Time [sec]','Fontsize',xaxis_sz,'interpreter','latex')
        ylabel('Range Residuals [m]','Fontsize',yaxis_sz,'interpreter','
           latex')
        scatter(data.hw3_p1_data(:,1),del_z(:,1),150,'*','DisplayName',
           legend_txt)
        legend('FontSize',legend_sz,'interpreter','latex','location','
           southeast')

        figure(2); hold on; grid on; set(gcf, 'WindowState', 'maximized')
           ;
        xlabel('Time [sec]','Fontsize',xaxis_sz,'interpreter','latex')
        ylabel('Range-Rate Residuals [m/s]','Fontsize',yaxis_sz,'
           interpreter','latex')
        scatter(data.hw3_p1_data(:,1),del_z(:,2),150,'*','DisplayName',
           legend_txt)
        legend('FontSize',legend_sz,'interpreter','latex','location','
           northeast')
    end
end

clear;

%% Problem 5: Angles-Only OD

% DATA PROVIDED ARE:
% T = (m x 1) measurement times [s]
% Z = (2 x m) RA/DEC measurements [deg]
% Pvv = (2 x 2 x m) RA/DEC measurement noise covariances [deg^2]
% Xobsv = (6 x m) pos. and vel. of the observer [km and km/s]
% Xtrue = (6 x m) true pos. and vel. of the object [km and km/s]

mu_E = 3.986004415e5; %[km^3/s^2]
data = load('data_HW03.mat');
opts = odeset('RelTol',1e-10,'AbsTol',1e-12);

% Plotting
plot_flag = false;
xaxis_sz = 20; yaxis_sz = 20; legend_sz = 18;
```

```
% Part a
% x_star = [32500;24500;2000;-2;2.5;.2];

% Part b
x_star = [35000;25000;1000;-1;3;1];

del_x0 = zeros(6,1);

n = 5;
MD_array = zeros(1,n);
error_array = zeros(6,n);
for i = 1:n
    CapLambda = zeros(6,6);
    Lambda = zeros(6,1);
    % Propogate states and STM for full time using current estimate of
       initial x
    tspan = data.T';
    Phi_0 = eye(6);
    x_full_0 = [x_star; reshape(Phi_0,36,1)];
    [~,x_full] = ode45(@(t,x) TwoBodyProp_STM(t,x,mu_E), tspan, x_full_0,
        opts);

    t0 = 0;
    h_array = zeros(length(data.Z),2);
    e_array = zeros(length(data.Z)*2,1);
    samples_e = zeros(length(data.Z),2);
    for j = 1:length(data.Z)
        t_l = data.T(j);
        z_l = deg2rad(data.Z(:,j));
        Pvv_l = data.Pvv(:,:,j)/(180^2/pi^2);

        x_l_star = x_full(j,1:6);
        Phi = reshape(x_full(j,7:42)',6,6);

        x_rel = x_l_star - data.Xobsv(:,j)';
        H =  H_tilde_fun(x_rel);
        x = x_rel(1);
        y = x_rel(2);
        z = x_rel(3);
        h = [(atan(y/x)); (atan(z/(sqrt(x^2+y^2))))];
        A = H*Phi;
        B = z_l - h;
        Lambda = Lambda + (A)'*(Pvv_l^-1)*(B);
        CapLambda = CapLambda + (A)'*(Pvv_l^-1)*(A);
        h_array(j,:) = h;
        e_array((2*j-1):(2*j),:) = B';
        samples_e(j,:) = B';
    end
    Pxx0 = CapLambda^-1;
    del_x0_hat = CapLambda\Lambda;
    x_star = x_star + del_x0_hat;
    del_x0 = del_x0 - del_x0_hat;
```

```
    disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
    disp(['The reference state after ', num2str(i), ' iterations is:'])
    x_star
    error_state = x_star - data.Xtrue(:,1);
    disp(['The Mahalanobis distance after ', num2str(i), ' iterations is:'])
    MD = sqrt(error_state'*Pxx0^-1*error_state)
    MD_array(1,i) = MD;

    W = eye(2*length(data.Z))*Pvv_l(1,1);
    Cost = e_array'*(W^-1)*e_array

    RMS_avg = (sqrt((sum(samples_e(:,1).^2))/length(data.Z)) + sqrt((sum(samples_e(:,2).^2))/length(data.Z)))/2

    del_z = deg2rad(data.Z)' - h_array;
    legend_txt = ['Iteration ',num2str(i)];

    if plot_flag
        figure(3); hold on; grid on; set(gcf, 'WindowState', 'maximized');
        xlabel('Time [sec]','Fontsize',xaxis_sz,'interpreter','latex')
        ylabel('Right-Ascension Residuals [rad]','Fontsize',yaxis_sz,'interpreter','latex')
        scatter(data.T,del_z(:,1),120,'*','DisplayName',legend_txt)
        legend('FontSize',legend_sz,'interpreter','latex','location','southeast')

        figure(4); hold on; grid on; set(gcf, 'WindowState', 'maximized');
        xlabel('Time [sec]','Fontsize',xaxis_sz,'interpreter','latex')
        ylabel('Declination Residuals [rad]','Fontsize',yaxis_sz,'interpreter','latex')
        scatter(data.T,del_z(:,2),120,'*','DisplayName',legend_txt)
        legend('FontSize',legend_sz,'interpreter','latex','location','northeast')
    end

    error_array(:,i) = error_state;

end
if plot_flag

    figure(5); hold on; grid on; set(gcf, 'WindowState', 'maximized');
    subplot(6,1,1)
    plot(linspace(1,n,n),error_array(1,:),'.-','LineWidth',2,'MarkerSize',20)
    ylabel('x [km]','Fontsize',yaxis_sz,'interpreter','latex')
    title('\textbf{State Estimation Error vs. Iteration}','Fontsize',25,'interpreter','latex')

    subplot(6,1,2)
```

```
    plot(linspace(1,n,n),error_array(2,:),'.-','LineWidth',2,'MarkerSize'
        ,20)
    ylabel('y [km]','Fontsize',yaxis_sz,'interpreter','latex')

    subplot(6,1,3)
    plot(linspace(1,n,n),error_array(3,:),'.-','LineWidth',2,'MarkerSize'
        ,20)
    ylabel('z [km]','Fontsize',yaxis_sz,'interpreter','latex')

    subplot(6,1,4)
    plot(linspace(1,n,n),error_array(4,:),'.-','LineWidth',2,'MarkerSize'
        ,20)
    ylabel('$v_x$ [$\frac{km}{s}$]','Fontsize',yaxis_sz,'interpreter','
        latex')

    subplot(6,1,5)
    plot(linspace(1,n,n),error_array(5,:),'.-','LineWidth',2,'MarkerSize'
        ,20)
    ylabel('$v_y$ [$\frac{km}{s}$]','Fontsize',yaxis_sz,'interpreter','
        latex')

    subplot(6,1,6)
    plot(linspace(1,n,n),error_array(6,:),'.-','LineWidth',2,'MarkerSize'
        ,20)
    xlabel('Iterations','Fontsize',xaxis_sz,'interpreter','latex')
    ylabel('$v_z$ [$\frac{km}{s}$]','Fontsize',yaxis_sz,'interpreter','
        latex')

    figure(6); hold on; grid on; set(gcf, 'WindowState', 'maximized');
    xlabel('Iterations','Fontsize',xaxis_sz,'interpreter','latex')
    ylabel('Mahalanobis distance','Fontsize',yaxis_sz,'interpreter','
        latex')
    plot(linspace(1,n,n),MD_array,'.-','LineWidth',3,'MarkerSize',30)
end
%% Functions

function [H_tilde] =  H_tilde_fun(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;

H_tilde = [-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_STM(~,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);
```

```
% Extract STM & reshape
Phi = x(7:end);
Phi = reshape(Phi,6,6);

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

% Multiply Phi by F
result = F*Phi;

% Reshape back to column
result = reshape(result,36,1);

dx(7:42,1) = result;

end
```