# Homework 7 Report

AERO626, Spring 2023

Name: David van Wijk  UIN: 932001896

## Problem 1 Results

A Gaussian mixture filter (GMF) was developed to estimate the position and velocity, using a set of four initial weights and covariances. In Figure 1, the state estimation error of the GMF versus measurement number can be found. The $3\sigma$ interval of state covariance is also plotted. Note that both the prior and posterior errors and covariances are plotted. The state errors stay well within the $3\sigma$ interval throughout the process, which is a fair visual indicator that the filter is functioning well. The final estimated values of both the position and velocity are quite close to the true position and velocity values. Since a GMF is a weighted sum of individual Gaussians, the mean and covariances that are plotted in Figure 1 are a representation of the overall Gaussian, using the definitions on page 539 and 541 of the course notes. We do not observe the interesting case of the uncertainty *increasing* with a measurement in this problem.

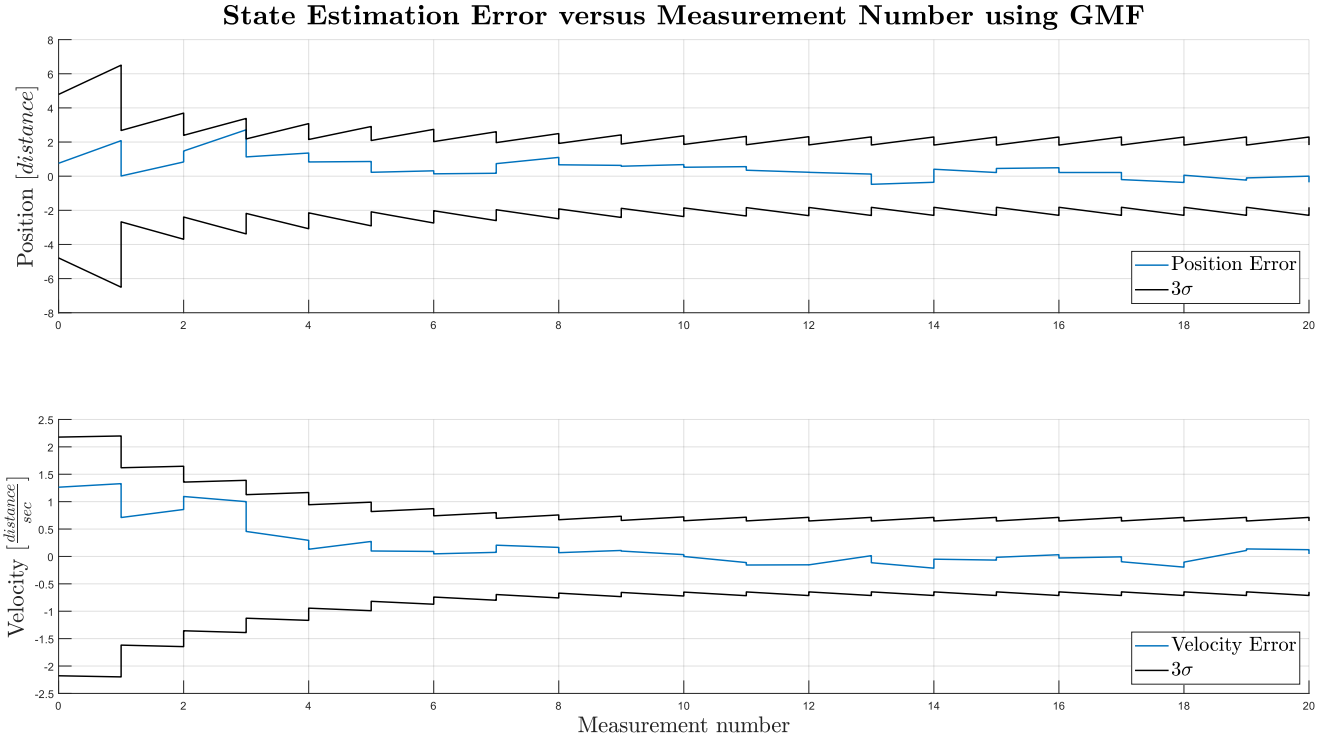


**Figure 1: State Estimation Error versus Measurement Noise using Gaussian Mixture Filter.**

In order to gain further insight in the performance of the GMF, it is useful to plot the weights of the individual Gaussians versus measurement number (Figure 2). Here, we see that almost immediately, the filter is able to discern that it is highly likely that the data came from the fourth Gaussian, and thus assigns a very high weight to that particular Gaussian (w4 in this case). Since the initial state value is known to us, we can confirm that indeed, the data came from the fourth Gaussian. Towards the end of the process, the remainder of the weights are nearly zero.

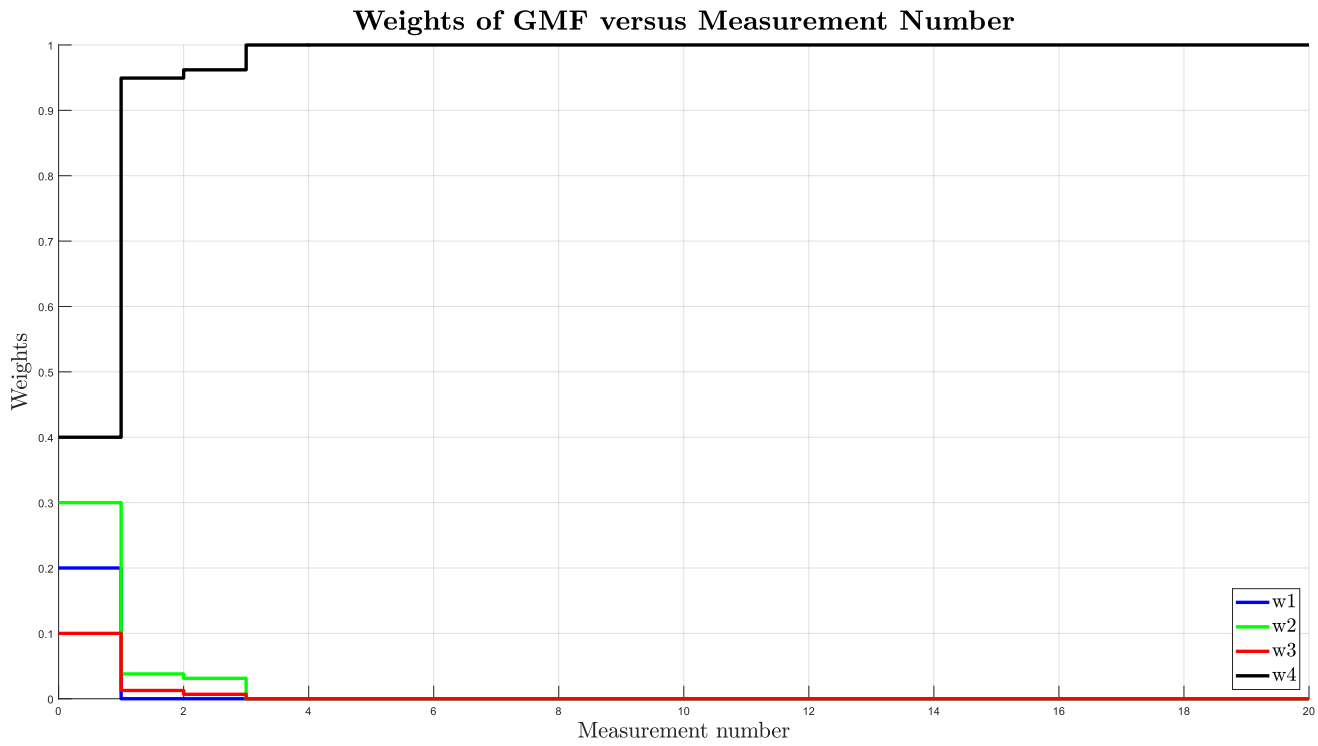


**Figure 2: Gaussian Mixture Filter Weights versus Measurement Number.**

The code for these results can be found in Appendix A.

## Problem 2 Results

The derivation for problem 2 can be found after the appendix.

# A Matlab Code

```
%% AERO 626 Homework 7
%
%   Texas A&M University
%   Aerospace Engineering
%   van Wijk, David

close all; clear; clc;
plot_flag = true;
xaxis_sz = 20; yaxis_sz = 20; legend_sz = 18;

% Seed to reproduce results
% rng(10)

%% Part A - Initialization and Data Processing

% Load in the provided data
data    = load('data_HW07.mat');
x0      = data.x0;
numPts  = 20;
x_truth = data.xk(:,1:numPts);
z_full  = data.zk(:,1:numPts);

% Define the dynamics and measurement Jacobians (constant)
Fx = [1 1; 0 1];
Hx = [1 0];

% Process noise and measurement noise covariances
Pww = diag([.01,.01]);
Pvv = 1;

% Define weights, means and covariances for the components of the GM
w1 = .2; m1x0 = [-2.5; -.5]; P1xx0 = diag([.2,.1]);
w2 = .3; m2x0 = [-1;    .2]; P2xx0 = diag([.25,.05]);
w3 = .1; m3x0 = [-.5;  -.3]; P3xx0 = diag([.2,.1]);
w4 = .4; m4x0 = [1.2;    1]; P4xx0 = diag([1,.3]);
w_full   = {w1,w2,w3,w4};
m_full   = {m1x0,m2x0,m3x0,m4x0};
Pxx_full = {P1xx0,P2xx0,P3xx0,P4xx0};

% Obtain initial mean and covarince
[mx0, Pxx0] = getMeanCovfromGM(w_full,m_full,Pxx_full);

% 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

% declare storage space for saving state estimation error information
xcount  = 0;
xstore  = nan(2,2*numPts-1);
kxstore = nan(1,2*numPts-1);
mxstore = nan(2,2*numPts-1);
exstore = nan(2,2*numPts-1);
sxstore = nan(2,2*numPts-1);
wxstore = nan(4,2*numPts-1);

% declare storage space
zcount  = 0;
zstore  = nan(2,numPts);
kzstore = nan(1,numPts);

% store initial data
xcount             = xcount + 1;
kxstore(:,xcount) = 0;
exstore(:,xcount) = x0 - mx0;
mxstore(:,xcount) = mx0;
sxstore(:,xcount) = sqrt(diag(Pxx0));
wxstore(:,xcount) = cell2mat(w_full(:));

for k = 1:numPts
    zk = z_full(k);     % current measurement to process
    Pvvk = Pvv;         % measurement noise covariance

    % unpack the truth -- this cannot be used in the filter, only for
       analysis
    xk = x_truth(:,k);  % true state

    % propagate means and covariances of GM
    [wxkm_full, mxkm_full, Pxxkm_full] = propagateMain(w_full,m_full,
       Pxx_full,Fx,Pww);

    % get full mean and covariance of GM
    [mxGM_prior, PxxGM_prior] = getMeanCovfromGM(wxkm_full,mxkm_full,
       Pxxkm_full);
    xcount             = xcount + 1;
    xstore(:,xcount)  = xk;
    kxstore(:,xcount) = k;
    mxstore(:,xcount) = mxGM_prior;
    exstore(:,xcount) = xk - mxGM_prior;
    sxstore(:,xcount) = sqrt(diag(PxxGM_prior));
    wxstore(:,xcount) = cell2mat(wxkm_full(:));

    % update means and covariances of GM
    [wxkp_full, mxkp_full, Pxxkp_full] = updateMain(wxkm_full,mxkm_full,
       Pxxkm_full,Hx,zk,Pvv);

    % get full mean and covariance of GM
```

```
    [mxGM_posterior, PxxGM_posterior] = getMeanCovfromGM(wxkp_full,
       mxkp_full,Pxxkp_full);

    % store a posteriori state information for analysis
    xcount            = xcount + 1;
    xstore(:,xcount)  = xk;
    kxstore(:,xcount) = k;
    mxstore(:,xcount) = mxGM_posterior;
    exstore(:,xcount) = xk - mxGM_posterior;
    sxstore(:,xcount) = sqrt(diag(PxxGM_posterior));
    wxstore(:,xcount) = cell2mat(wxkp_full(:));

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

    w_full   = wxkp_full;
    m_full   = mxkp_full;
    Pxx_full = Pxxkp_full;
end

if plot_flag
    plotEstimationError(exstore,sxstore,kxstore,xaxis_sz,yaxis_sz,
       legend_sz)
    plotWeights(wxstore,kxstore,xaxis_sz,yaxis_sz,legend_sz)
end

%% Functions

function [wxkm_full, mxkm_full, Pxxkm_full] = propagateMain(w_full,m_full
   ,Pxx_full,Fx,Pww)

mxkm_full  = cell(1,length(m_full));
Pxxkm_full = cell(1,length(m_full));
for i = 1:length(m_full)
    % propagage mean
    mxkm1p = m_full{i};
    mxkm = Fx*mxkm1p;
    mxkm_full{i} = mxkm;

    % propagate covariance
    Pxxkm1p = Pxx_full{i};
    Pxxkm = Fx*Pxxkm1p*Fx' + Pww;
    Pxxkm_full{i} = Pxxkm;
end
wxkm_full = w_full;

end

function [wxkp_full, mxkp_full, Pxxkp_full] = updateMain(wxkm_full,
   mxkm_full,Pxxkm_full,Hx,zk,Pvv)
```

```
mxkp_full  = cell(1,length(mxkm_full));
Pxxkp_full = cell(1,length(mxkm_full));
wxkp_full  = cell(1,length(mxkm_full));
kk_full    = cell(1,length(mxkm_full));
wk_sum     = 0;
for i = 1:length(mxkm_full)
    mxkm  = mxkm_full{i};
    Pxxkm = Pxxkm_full{i};
    wxkm  = wxkm_full{i};
    Pzzkm = Hx*Pxxkm*Hx' + Pvv;
    mzkm  = Hx*mxkm;

    % compute baby Kalman gain
    kk_l       = abs(2*pi*Pzzkm)^-.5*exp(-.5*(zk - mzkm)'*Pzzkm^-1*(zk -
       mzkm));
    kk_full{i} = kk_l;
    wk_sum     = wk_sum + kk_l*wxkm;
end

for i = 1:length(mxkm_full)
    mxkm  = mxkm_full{i};
    Pxxkm = Pxxkm_full{i};
    wxkm  = wxkm_full{i};
    Pzzkm = Hx*Pxxkm*Hx' + Pvv;
    mzkm  = Hx*mxkm;

    % update weights
    kk_l = kk_full{i};
    wxkp = (kk_l*wxkm)/wk_sum;
    wxkp_full{i} = wxkp;

    % compute Kalman gain
    Kk = (Pxxkm*Hx')/(Pzzkm);

    % update mean
    mxkp = mxkm + Kk*(zk - mzkm);
    mxkp_full{i} = mxkp;

    % propagate covariance
    Pxkkp = Pxxkm - Kk*Hx*Pxxkm;
    Pxxkp_full{i} = Pxkkp;
end

end

function [mxGM, PxxGM] = getMeanCovfromGM(wx_full,mx_full,Pxx_full)
mxGM = 0;
for j = 1:length(wx_full)
    mx_j  = mx_full{j};
    w_j   = wx_full{j};
    mxGM  = mxGM + w_j*mx_j;
end
```

```
PxxGM = zeros(size(Pxx_full{1}));
for j = 1:length(wx_full)
    mx_j   = mx_full{j};
    w_j    = wx_full{j};
    Pxx_j  = Pxx_full{j};
    PxxGM  = PxxGM + w_j*(Pxx_j + (mx_j - mxGM)*(mx_j - mxGM)');
end
end

%% Plotting Functions

function plotEstimationError(exstore,sxstore,kxstore,xaxis_sz,yaxis_sz,
   legend_sz)
std_plot = 3; txt = [num2str(std_plot) '$\sigma$'];

err_line_opts = {'-','LineWidth',1.3};
std_line_opts = {'-','LineWidth',1.3,'Color','k'};

figure; grid on; set(gcf, 'WindowState', 'maximized');
subplot(2,1,1); hold on; grid on;
title('\textbf{State Estimation Error versus Measurement Number using GMF
   }','Fontsize',25,'interpreter','latex')
a1 = plot(kxstore,exstore(1,:),err_line_opts{:});
a2 = plot(kxstore,std_plot*sxstore(1,:),std_line_opts{:});
plot(kxstore,-std_plot*sxstore(1,:),std_line_opts{:})
ylabel('Position [$distance$]','Fontsize',yaxis_sz,'interpreter','latex')
legendtxt = {'Position Error',txt};
legend([a1 a2],legendtxt,'FontSize',legend_sz,'interpreter','latex','
   location','southeast')

subplot(2,1,2); hold on; grid on;
a1 = plot(kxstore,exstore(2,:),err_line_opts{:});
a2 = plot(kxstore,std_plot*sxstore(2,:),std_line_opts{:});
plot(kxstore,-std_plot*sxstore(2,:),std_line_opts{:})
ylabel('Velocity [$\frac{distance}{sec}$]','Fontsize',yaxis_sz,'
   interpreter','latex')
xlabel('Measurement number','Fontsize',xaxis_sz,'interpreter','latex')
legendtxt = {'Velocity Error',txt};
legend([a1 a2],legendtxt,'FontSize',legend_sz,'interpreter','latex','
   location','southeast')

end

function plotWeights(wxstore,kxstore,xaxis_sz,yaxis_sz,legend_sz)

w1_line_opts = {'-','LineWidth',3,'Color','b'};
w2_line_opts = {'-','LineWidth',3,'Color','g'};
w3_line_opts = {'-','LineWidth',3,'Color','r'};
w4_line_opts = {'-','LineWidth',3,'Color','k'};

figure; set(gcf, 'WindowState', 'maximized');
hold on; grid on;
```

```
title('\textbf{Weights of GMF versus Measurement Number}','Fontsize',25,'
    interpreter','latex')
a1 = plot(kxstore,wxstore(1,:),w1_line_opts{:});
a2 = plot(kxstore,wxstore(2,:),w2_line_opts{:});
a3 = plot(kxstore,wxstore(3,:),w3_line_opts{:});
a4 = plot(kxstore,wxstore(4,:),w4_line_opts{:});
ylabel('Weights','Fontsize',yaxis_sz,'interpreter','latex')
legendtxt = {'w1','w2','w3','w4'};
legend([a1 a2 a3 a4],legendtxt,'FontSize',legend_sz,'interpreter','latex'
    ,'location','southeast')
xlabel('Measurement number','Fontsize',xaxis_sz,'interpreter','latex')

end
```