HW - 4 Submission

AE-6505 - Kalman Filtering

Kadhir Umasankar

Spring 2022

03-28-22 - HW 4

Sunday, March 27, 2022 6:19 PM

5.4. a. Representing the equations in state space form

$$\begin{bmatrix} x_{p,k+1} \\ x_{g,k+1} \end{bmatrix} = \begin{bmatrix} 1-k_2 & k_1 \\ -k_3 & 1 \end{bmatrix} \begin{bmatrix} x_{p,k} \\ x_{g,k} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v_k + \begin{bmatrix} \omega_{p,k} \\ \omega_{g,k} \end{bmatrix}$$

$$= \begin{bmatrix} 1/2 & 1 \\ -1/2 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v_k + \omega_k$$

$$y_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{p,k} \\ x_{g,k} \end{bmatrix} + v_k$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x_k + v_k$$

From info in the question, we also know that

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

So $\omega_k \sim N(0, Q)$ and $v_k \sim N(0, R)$

b. Given that initial population count is perfect

$$\therefore P_0^+ = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Iteratively using the Kalman Filter equations

$$P_1^- = FP_0^+F^T + Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$K_1 = P_1^- H^T (HP_1^- H^T + R)^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1/2 \end{bmatrix}$$

$$P_1^+ = (I - K_1 H) P_1^- = \begin{bmatrix} 0 & 0 \\ 0 & 1/2 \end{bmatrix}$$

$$P_2^- = FP_1^+ F^T + Q = \begin{bmatrix} 3/2 & 1/2 \\ 1/2 & 3/2 \end{bmatrix}$$

$$K_2 = P_2^- H^T (HP_2^- H^T + R)^{-1} = \begin{bmatrix} 1 & 0 \\ 1/7 & 4/7 \end{bmatrix}$$

$$K_2 = P_2^- H^T (H P_2^- H^T + R)^{-1} = \begin{bmatrix} 1 & 0 \\ 1/7 & 4/7 \end{bmatrix}$$

$$P_2^+ = (I - K_2 H) P_2^- = \begin{bmatrix} 0 & 0 \\ 0 & 4/7 \end{bmatrix}$$

∴ After 1 week the variance of the guppy population is $1/2$ and $4/7$ after two weeks

c. As $k \to \infty$ $x_{k+1} = x_k$ (i.e. steady-state)

$$x_k = F x_k + G u_k$$
$$(I - F) x_k = G u_k$$
$$x_k = (I - F)^{-1} G u_k$$
$$= \begin{bmatrix} 2 \\ 1 \end{bmatrix} u_k$$

∴ The ratio of the piranha population to the guppy population is 2:1

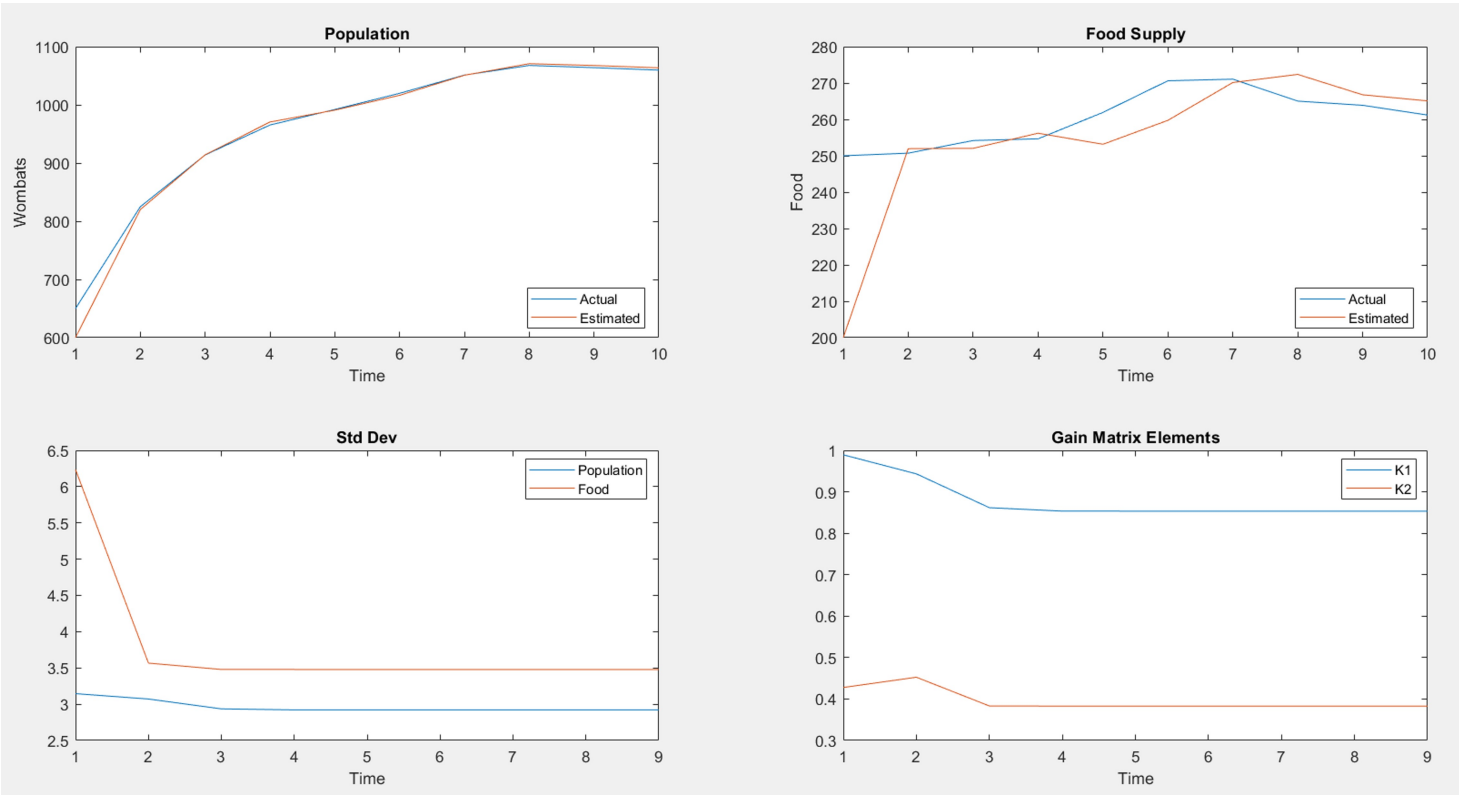
5.11. $$\begin{bmatrix} p_{k+1} \\ f_{k+1} \end{bmatrix} = \begin{bmatrix} 1/2 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_k \\ f_k \end{bmatrix} + \omega_k$$

$$\omega_{f,k} \sim N(0, Q) \quad \text{where} \quad Q = \begin{bmatrix} 0 & 0 \\ 0 & 10 \end{bmatrix}$$

$$y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} p_k \\ f_k \end{bmatrix} + v_k$$

$$v_k \sim N(0, R) \quad \text{where} \quad R = 10$$

After assembling this in MATLAB and using Eq (5.17) through (5.19) from Simon, we get the following plots

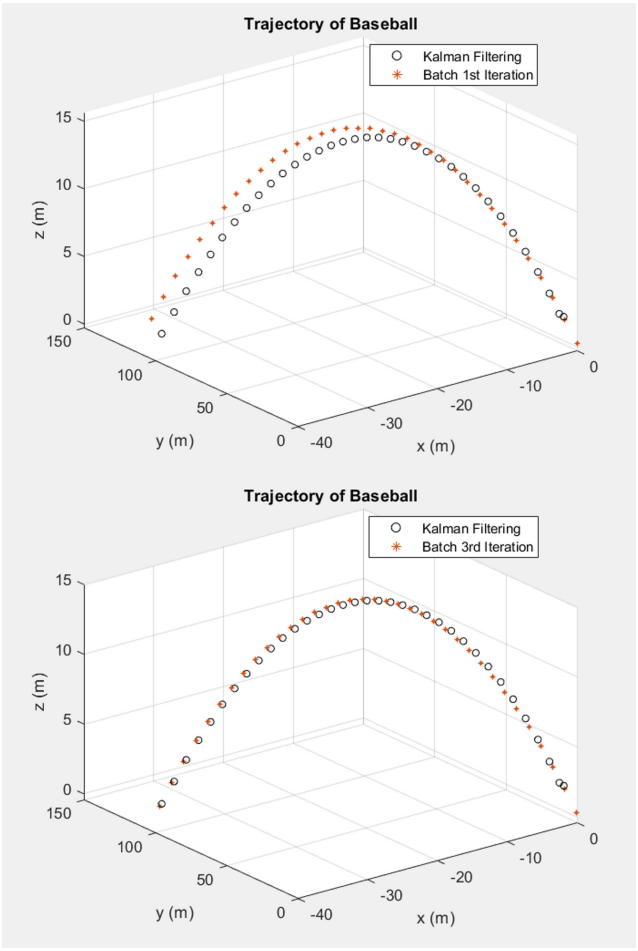


b. The std. devs of the population and food supply estimation errors starts around 15. The steady state std. devs. settle at ~2.92 for the wombat population and ~3.47 for the food supply. This difference is due to the large error in the initial estimate and the small number of simulation iterations.

c. As the number of timesteps increases, the std dev. moves closer to the theoretical std. dev because the initial estimation errors are dealt with as more simulations are run.

2. Made necessary changes to batch approach for a priori information
   - Implemented Kalman Filter code

   (Code in appendix)


Trajectory of Baseball
Kalman Filtering
Batch 1st Iteration
z (m)
y (m)
x (m)
Trajectory of Baseball
Kalman Filtering
Batch 3rd Iteration
z (m)
y (m)
x (m)

1 Appendix: MATLAB Code

```
%%
clc; clear all; close all;
maxIter = 10;
foodW = normrnd(0,sqrt(10),[1,maxIter]);
obsW = normrnd(0,sqrt(10),[1,maxIter]);
Xactual = [650; 250];
F = [1/2 2; 0 1];
Pplus(:,:,1) = [500 0; 0 200];
Q = [0 0; 0 10];
H = [1 0];
R = 10;
Xplus(:,1)= [600; 200];
n = 2;

y(:,1) = Xactual(1,1) + obsW(1);
std1 = [];
std2 = [];
K1 = [];
K2 = [];

% discrete kalman filter + simulation
for k = 2:maxIter
     Xactual(:,k) = F*Xactual(:,k-1) + [0 foodW(k)]';
     y(:,k) = Xactual(1,k) + obsW(k);

     Pminus(:,:,k) = F*Pplus(:,:,k-1)*F' + Q;
     K(:,:,k) = Pminus(:,:,k)*H'/(H*Pminus(:,:,k)*H' + R);
     Xminus(:,k) = F*Xplus(:,k-1);
     Xplus(:,k) = Xminus(:,k) + K(:,:,k)*(y(:,k)-H*Xminus(:,k));
     Pplus(:,:,k) = (eye(n) - K(:,:,k)*H)*Pminus(:,:,k);
     std1 = vertcat(std1, sqrt(Pplus(1,1,k)));
     std2 = vertcat(std2, sqrt(Pplus(2,2,k)));
     K1 = vertcat(K1, K(1,1,k));
     K2 = vertcat(K2, K(2,1,k));
end
figure()
subplot(2,2,1)
plot(Xactual(1,:))
hold on
plot(Xplus(1,:))
xlabel('Time')
ylabel('Wombats')
legend('Actual', 'Estimated', 'Location', 'best')
title('Population')

subplot(2,2,2)
plot(Xactual(2,:))
hold on
plot(Xplus(2,:))
xlabel('Time')
ylabel('Food')
legend('Actual', 'Estimated', 'Location', 'best')
title('Food Supply')

subplot(2,2,3)
plot(std1)
hold on
plot(std2)
xlabel('Time')
legend('Population', 'Food', 'Location', 'best')
title('Std Dev')

```

```
subplot(2,2,4)
plot(K1)
hold on
plot(K2)
xlabel('Time')
legend('K1', 'K2', 'Location', 'best')
title('Gain Matrix Elements')

%%
%%
clc; clear all; close all;

% SET CONSTANTS
g = 9.81;

% Measurement noise covariance
% 5 ft and 0.1 deg in m and rad
R = [(1.524)^2 0 0;
     0 (0.1*pi/180)^2 0;
     0 0 (0.1*pi/180)^2];

data = importdata('homerun_data_HW4.txt');
data(1,:) = [];
tspan = data(:, 1);
rho = data(:, 2)/3.281; % in m
alpha = data(:, 3)*pi/180; % in rad
beta = data(:, 4)*pi/180; % in rad

% Assembling Y_obs using read in data
Y_obs = [];
for i = 1:length(rho)
    Y_obs = vertcat(Y_obs, [rho(i) alpha(i) beta(i)]);
end
m = size(Y_obs, 1);

% Visualizing data
figure(1)
[x y z] = sph2cart(alpha, beta, rho);
scatter3(x, y, z, 10, 'm', 'filled')

t0 = 0;
X0star = [0.4921 0.4921 2.0013 -26.2467 114.3051 65.9941]'/3.281; % in SI
xbar_0 = [0 0 0 0 0 0]';
Pbar_0 = diag([4 4 4 0.1 0.1 0.1])/3.281^2;
i = 1;

while i<=m
    if i == 1
        xhat_i_1 = X0star;
        P_i_1 = Pbar_0;
        t_i_1 = t0;
    else
        xhat_i_1 = xhat_i;
        P_i_1 = P_i;
        t_i_1 = tspan(i-1);
    end

    Phi = [1 0 0 tspan(i)-t_i_1 0 0;
           0 1 0 0 tspan(i)-t_i_1 0;
           0 0 1 0 0 tspan(i)-t_i_1;
           0 0 0 1 0 0;
           0 0 0 0 1 0;
           0 0 0 0 0 1];

    B = [0; 0; -1/2*(tspan(i)-t_i_1)^2; 0; 0; -(tspan(i)-t_i_1)];
```

```
    xbar_i = Phi * xhat_i_1 + B * g;
    Pbar_i = Phi * P_i_1 * Phi';
%     + Q to previous line to add filter from divering

    X = xbar_i(1);
    Y = xbar_i(2);
    Z = xbar_i(3);

    rho = sqrt(X^2 + Y^2 + Z^2);
    alpha = atan2(Y, X);
    beta = atan2(Z, sqrt(X^2 + Y^2));

    Y_computed = [rho; alpha; beta];
    y_i = Y_obs(i,:)' - Y_computed;
    H_tilda_i = [X/rho Y/rho Z/rho 0 0 0;
                 (-Y/X^2)/(1 + (Y/X)^2) (1/X)/(1+(Y/X)^2) 0 0 0 0;
                 ((-X*Z)/(X^2 + Y^2)^(3/2))/(1+(Z^2)/(X^2 + Y^2)) ((-Y*Z)/(X^2 + Y^2)^(3/2))/(1 + (Z^2)/(X^2
↪  + Y^2)) ((1)/(X^2 + Y^2)^(1/2))/(1 + (Z^2)/(X^2 + Y^2)) 0 0 0];
    K_i = Pbar_i*H_tilda_i'*inv(H_tilda_i*Pbar_i*H_tilda_i' + R);

    xhat_i = xbar_i + K_i*(y_i);
    P_i = (eye(6) - K_i*H_tilda_i)*Pbar_i;
    X_calculated(:, i) = xhat_i;

    figure(1)
    hold on
    scatter3(X_calculated(1, i), X_calculated(2, i), X_calculated(3, i), 20, 'black')

    i = i+1;
end

xlabel('x (m)')
ylabel('y (m)')
zlabel('z (m)')
title('Trajectory of Baseball')

%% Batch approach
% Hyperparameters
n = 6; % number of params
k = 3; % number of obs per epoch
max_iter = 8;
g = 9.81; % m/s^2
dt = 0.1;
t_all = [0:dt:3.5];

% Measurement noise covariance
% 5 ft and 0.1 deg in m and rad
R = [(1.524)^2 0 0;
     0 (0.1*pi/180)^2 0;
     0 0 (0.1*pi/180)^2];

% Initial state
X_0 = [0.4921 0.4921 2.0013 -26.2467 114.3051 65.9941]'; % in ft and ft/s
X_0 = X_0/3.281; % converting to SI

% IF NO APRIORI COMMENT THIS OUT
P_bar = diag([4 4 4 0.1 0.1 0.1])/3.281^2;
x_bar = [0; 0; 0; 0; 0; 0];

% Initial measurements
data = importdata('../HW3/homerun_data.txt');
tspan = data(:, 1);
rho = data(:, 2)/3.281; % in m
alpha = data(:, 3)*pi/180; % in rad
beta = data(:, 4)*pi/180; % in rad
```

```
192
193 % Assembling Y_obs using read in data
194 Y_obs = [];
195 for i = 1:length(rho)
196     Y_obs = vertcat(Y_obs, [rho(i) alpha(i) beta(i)]);
197 end
198 m = size(Y_obs, 1);
199
200 % Visualizing data
201 figure(2)
202 [x y z] = sph2cart(alpha, beta, rho)
203 scatter3(x, y, z, 10, 'm', 'filled')
204
205 % Run iterations
206 for i = 1:max_iter
207     HtH = zeros(n, n);
208     Hty = zeros(n, 1);
209
210     % Loop over all observations
211     % Compute the nominal trajectory and computed observations
212     % Compute Normal equations
213     for j=1:m
214         Phi = [1 0 0 tspan(j) 0 0;
215                0 1 0 0 tspan(j) 0;
216                0 0 1 0 0 tspan(j);
217                0 0 0 1 0 0;
218                0 0 0 0 1 0;
219                0 0 0 0 0 1];
220
221         % Compute nominal trajectory
222         B = [0; 0; -1/2*tspan(j)^2; 0; 0; -tspan(j)];
223         X_nom = Phi * X_0 + B * g;
224
225         X = X_nom(1);
226         Y = X_nom(2);
227         Z = X_nom(3);
228
229         % Compute partials
230         rho = sqrt(X^2 + Y^2 + Z^2);
231
232         H_tilda = [X/rho Y/rho Z/rho 0 0 0;
233                    (-Y/X^2)/(1 + (Y/X)^2) (1/X)/(1+(Y/X)^2) 0 0 0 0;
234                    ((-X*Z)/(X^2 + Y^2)^(3/2))/(1+(Z^2)/(X^2 + Y^2)) ((-Y*Z)/(X^2 + Y^2)^(3/2))/(1 +
↪  (Z^2)/(X^2 + Y^2)) ((1)/(X^2 + Y^2)^(1/2))/(1 + (Z^2)/(X^2 + Y^2)) 0 0 0];
235
236         % Generate observation deviation (i.e. observed - computed)
237         rho = sqrt(X^2 + Y^2 + Z^2);
238         alpha = atan2(Y, X);
239         beta = atan2(Z, sqrt(X^2 + Y^2));
240
241         Y_computed = [rho; alpha; beta];
242         y = Y_obs(j,:)' - Y_computed;
243
244         % Use STM to map partials back to initial coords
245         H = H_tilda * Phi;
246
247         % Compute rank-k update of normal equations for this observation
248         HtH = HtH + H'*inv(R)*H;
249         Hty = Hty + H'*inv(R)*y;
250     end
251
252     % Compute state deviation
253     P = inv(HtH);
254
255     % Uncomment line 1 if no apriori, uncomment line 2 if given apriori info
```

```
%     x_hat = P * Hty;
    x_hat = inv(HtH + inv(P_bar))*(Hty + inv(P_bar)*x_bar);

    % Add state deviation to get final estimate
    X_new = X_0 + x_hat

    % Use the new estimates as starting point for next iter
    X_0 = X_new;
    P_bar = inv(HtH + inv(P_bar));

    %Plot estimated trajectory for current iteration
    figure(2);
    hold on;
    if i==1 || i == 3
        for k=1:size(t_all, 2)
            X0 = X_0(1);
            Y0 = X_0(2);
            Z0 = X_0(3);
            Vx0 = X_0(4);
            Vy0 = X_0(5);
            Vz0 = X_0(6);
            X = X0 + Vx0 * t_all(k);
            Y = Y0 + Vy0 * t_all(k);
            Z = Z0 + Vz0 * t_all(k) - (1/2)*g*t_all(k)^2;
            X_KF(i, k) = X;
            Y_KF(i, k) = Y;
            Z_KF(i, k) = Z;
            scatter3(X, Y, Z, 20)
        end
    end
end

xlabel('x (m)')
ylabel('y (m)')
zlabel('z (m)')
title('Trajectory of Baseball')

%%
close all;

figure()
scatter3(X_calculated(1,:), X_calculated(2,:), X_calculated(3,:), 20, 'black')
hold on;
scatter3(X_KF(1, :), Y_KF(1, :), Z_KF(1, :), 10, '*')
xlabel('x (m)')
ylabel('y (m)')
zlabel('z (m)')
title('Trajectory of Baseball')
legend('Kalman Filtering', 'Batch 1st Iteration', 'Location', 'best')
hold off;

figure()
scatter3(X_calculated(1,:), X_calculated(2,:), X_calculated(3,:), 20, 'black')
hold on;
scatter3(X_KF(3, :), Y_KF(3, :), Z_KF(3, :),  10, '*')
xlabel('x (m)')
ylabel('y (m)')
zlabel('z (m)')
title('Trajectory of Baseball')
legend('Kalman Filtering', 'Batch 3rd Iteration', 'Location', 'best')
hold off;
```