Exercise: design of Kalman predictors and filters for a LTI dynamic system

Consider the following LTI dynamic system $\mathcal{S}$:

$$
\begin{aligned}
x(t+1) &= Ax(t) + Bu(t) + v_1(t) \\
y(t) &= Cx(t) + Du(t) + v_2(t)
\end{aligned}
$$

where

$$
A = \begin{bmatrix} 0.96 & 0.5 & 0.27 & 0.28 \\ -0.125 & 0.96 & -0.08 & -0.07 \\ 0 & 0 & 0.85 & 0.97 \\ 0 & 0 & 0 & 0.99 \end{bmatrix}, \; B = \begin{bmatrix} 1 \\ -1 \\ 2 \\ 1 \end{bmatrix}, \; C = \begin{bmatrix} 0 & 2 & 0 & 0 \end{bmatrix}, \; D = \begin{bmatrix} 0 \end{bmatrix}
$$

$v_1(t)$ is a white noise with zero mean value and variance $V_1 = B_{v_1} B_{v_1}^T$, $B_{v_1} = \sqrt{15}\left[0.5 \; 0 \; 0 \; 1\right]^T$, $v_2(t)$ is a white noise with zero mean value and variance $V_2 = 2000$ and $u(t)$ is a suitable input signal whose $N = 4000$ values have been saved in the MATLAB `data.mat` file. The noises $v_1(t)$ and $v_2(t)$ are uncorrelated, i.e., $V_{12} = 0$. The initial state $x(1)$ is a random vector with zero mean value and variance $P_1 = E\left[x(1)x(1)^T\right] = 0.5 I_4$; in the following simulations, assume $x(1) = \begin{bmatrix} 30 & 40 & -70 & -10 \end{bmatrix}^T$ as realization.

Problem: Design the following predictors and filters:

- Dynamic Kalman 1-step predictor $\mathcal{K}$ in standard form;
- Dynamic Kalman filter $\mathcal{F}$ in standard form;
- Steady-state Kalman 1-step predictor $\mathcal{K}^\infty$ in standard form;
- Steady-state Kalman filter $\mathcal{F}^\infty$ in standard form;
- (Optional) Dynamic Kalman 1-step predictor $\mathcal{K}_{pc}$ in predictor/corrector form.

Evaluate the quality of the different predictors and filters, comparing by means of plots the obtained state and output estimates; evaluate the Root Mean Square Errors:

$$
\begin{aligned}
RMSE_{x_k} &= \sqrt{\frac{1}{N - N_0} \sum_{t=N_0+1}^{N} \left[x_k(t) - \hat{x}_k(t)\right]^2}, \quad k = 1, \ldots, 4 \\
RMSE_y &= \sqrt{\frac{1}{N - N_0} \sum_{t=N_0+1}^{N} \left[y(t) - \hat{y}(t)\right]^2}
\end{aligned}
$$

being $\hat{x}_k(t)$ and $\hat{y}(t)$ the estimates of the state $x_k(t)$ and the output $y(t)$, respectively, and compare the different results obtained for $N_0 = 0$, 20 and 100.

Main steps:

(1) Load the input u from the `data.mat` file (use `load` MATLAB command).

(2) Simulate the system $\mathcal{S}$ inside a `for` loop where, at each step $t = 1, 2, \ldots, N$:
the noise $v_2(t)$ is computed as `v2(t)=sqrt(V2)*randn` (MATLAB command `randn` generates random numbers v chosen from the univariate normal distribution with zero mean and variance $\sigma_v^2 = 1$, so that $w = \sqrt{V_2} v$ has variance $\Sigma_w = E[ww^T] = V_2$; in any case, verify at the end that the sampled variance of v_2 computed as `cov(v2')` approximates V_2);
the noise $v_1(t)$ is computed as `v1(:,t)=mvnrnd(zeros(1,n),Bv1*Bv1')'` (MATLAB command `mvnrnd(mu,Sigma)` generates random vectors $v \in \mathbb{R}^{1\times n}$ chosen from the multivariate normal distribution with mean `mu` $\in \mathbb{R}^{1\times n}$ and variance `Sigma` $\in \mathbb{R}^{n\times n}$; in any case, verify at the end that the sampled variance of v_1 computed as `cov(v1')` approximates V_1).
At the end, outside the `for` loop, plot the four states $x_k(t)$, $k = 1, \ldots, 4$, and the output $y(t)$ on different figures.
Remark: to produce the same random numbers at each program run, put the MATLAB command `rng('default')` at the beginning of the code.

(3) Simulate the Dynamic Kalman 1-step predictor $\mathcal{K}$ inside a second `for` loop:

$$\begin{cases} \hat{y}(t|t-1) = C\hat{x}(t|t-1) \\ e(t) = y(t) - \hat{y}(t|t-1) \\ K(t) = \left[AP(t)C^T + V_{12}\right]\left[CP(t)C^T + V_2\right]^{-1} \\ \hat{x}(t+1|t) = A\hat{x}(t|t-1) + Bu(t) + K(t)e(t) \\ P(t+1) = AP(t)A^T + V_1 - K(t)\left[CP(t)C^T + V_2\right]K(t)^T \end{cases}$$

initializing the variables as: $\hat{x}(1|0) = \mathbf{0}$, $P(1) = P_1$. At the end, outside the `for` loop, compute the $RMSE$s and add to the figures generated at step (2) the plots of the predicted states $\hat{x}_k(t|t-1)$, $k = 1, \ldots, 4$, and output $\hat{y}(t|t-1)$.

(4) Simulate the Dynamic Kalman filter $\mathcal{F}$ inside the second `for` loop:

$$\begin{cases} K_0(t) = P(t)C^T\left[CP(t)C^T + V_2\right]^{-1} \\ \hat{x}(t|t) = \hat{x}(t|t-1) + K_0(t)e(t) \\ \hat{y}(t|t) = C(t)\hat{x}(t|t) \end{cases}$$

where $\hat{x}(t|t-1)$ is the dynamic prediction provided by $\mathcal{K}$. At the end, outside the `for` loop, compute the $RMSE$s and add to the figures generated at step (2) the plots of the filtered states $\hat{x}_k(t|t)$, $k = 1, \ldots, 4$, and output $\hat{y}(t|t)$.

(5) Simulate the Steady-state Kalman 1-step predictor $\mathcal{K}^\infty$ inside a third `for` loop:

$$\begin{cases} \hat{y}^\infty(t|t-1) = C\hat{x}^\infty(t|t-1) \\ e^\infty(t) = y(t) - \hat{y}^\infty(t|t-1) \\ \hat{x}^\infty(t+1|t) = A\hat{x}^\infty(t|t-1) + Bu(t) + \bar{K}e^\infty(t) \end{cases}$$

initializing the variable $\hat{x}^\infty(1|0) = \mathbf{0}$ and computing $\bar{K}$ just before the `for` loop with the MATLAB command `kalman`. At the end, outside the `for` loop, compute the $RMSE$s and add to the figures generated at step (2) the plots of the predicted states $\hat{x}_k^\infty(t|t-1)$, $k = 1, \ldots, 4$, and output $\hat{y}^\infty(t|t-1)$.

(6) Simulate the Steady-state Kalman filter $\mathcal{F}^\infty$ inside the third `for` loop:

$$\begin{cases} \hat{x}^\infty(t|t) = \hat{x}^\infty(t|t-1) + \bar{K}_0 e^\infty(t) \\ \hat{y}^\infty(t|t) = C\hat{x}^\infty(t|t) \end{cases}$$

where $\hat{x}^\infty(t|t-1)$ is the steady-state prediction provided by $\mathcal{K}^\infty$ and $\bar{K}_0$ is given by the MATLAB command `kalman` (verify that $\bar{K} = A\bar{K}_0$). At the end, outside the `for` loop, compute the $RMSE$s and add to the figures generated at step (2) the plots of the filtered states $\hat{x}_k^\infty(t|t)$, $k = 1, \ldots, 4$, and output $\hat{y}^\infty(t|t)$.

(7) (Optional) Simulate the Dynamic Kalman 1-step predictor $\mathcal{K}_{pc}$ in predictor/corrector form inside a fourth `for` loop:

$$\begin{cases} K_0(t) = P(t)C^T\left[CP(t)C^T + V_2\right]^{-1} \\ P_0(t) = \left[I_n - K_0(t)C\right]P(t)\left[I_n - K_0(t)C\right]^T + K_0(t)V_2K_0(t)^T \\ \hat{y}_{pc}(t|t-1) = C\hat{x}_{pc}(t|t-1) \\ e_{pc}(t) = y(t) - \hat{y}_{pc}(t|t-1) \\ \hat{x}_{pc}(t|t) = \hat{x}_{pc}(t|t-1) + K_0(t)e_{pc}(t) \\ P(t+1) = AP_0(t)A^T + V_1 \\ \hat{x}_{pc}(t+1|t) = A\hat{x}_{pc}(t|t) + Bu(t) \end{cases}$$

initializing the variables as: $\hat{x}_{pc}(1|0) = \mathbf{0}$, $P(1) = P_1$.

Possible solution under MATLAB (file Lab4.m)

```
%% Laboratory 4 - Estimation, filtering and system identification - Prof. M. Taragna
% *Exercise: Design of Kalman predictors and filters for a LTI dynamic system*
%% Introduction
% The program code may be splitted in sections using the characters "%%".
% Each section can run separately with the command "Run Section"
% (in the Editor toolbar, just to the right of the "Run" button). You can do the
% same thing by highlighting the code you want to run and by using the button
% function 9 (F9). This way, you can run only the desired section of your code,
% saving your time. This script can be considered as a reference example.

clear all, close all, clc

%% Procedure
% # Load the file |data.mat| containing the input signal
% # Define the LTI dynamic system S
% # Define the noise variances
% # Set the initial state of the LTI dynamic system S
% # Simulate the LTI dynamic system S
% # Plot states and output of the LTI dynamic system S
% # Initialize the dynamic predictor K
% # Simulate the dynamic predictor K and the filter F
% # Compute the RMSEs for the dynamic predictor K and the filter F
% # Plot the estimated states and output versus the actual ones
% # Initialize the steady-state predictor Kinf
% # Simulate the steady-state predictor Kinf and the filter Finf
% # Compute the RMSEs for the steady-state predictor Kinf and the filter Finf
% # Plot the estimated states and output versus the actual ones
% # (Optional) Initialize the dynamic predictor Kpc
% # (Optional) Simulate the dynamic predictor Kpc

%% Problem setup

% Step 1: load of data

load data
% u = input signal, computed as: sign(sin(2*pi*0.0005*(1:4000)))*1+10;

N=length(u); % N = number of data
N0_vector=[0, 20, 100];

% Step 2: definition of LTI dynamic system S

A=[ 0.96,  0.5,   0.27,  0.28; ...
   -0.125, 0.96, -0.08, -0.07; ...
    0,     0,     0.85,  0.97; ...
    0,     0,     0,     0.99];
B=[1; -1; 2; 1];
C=[0, 2, 0, 0];
D=[0];
% Note that: A is stable, (A,C) is observable

% Step 3: definition of noise variances

Bv1=sqrt(15)*[0.5; 0; 0; 1]; % Note that: (A,Bv1) is reachable
V1=Bv1*Bv1';
V2=2000;
V12=0;
rng('default'); % To produce the same random numbers at each run

%% LTI dynamic system simulation
```

```
% Step 4: LTI dynamic system initialization

x(:,1)=[30; 40; -70; -10];
[n,nn]=size(A);

% Step 5: LTI dynamic system simulation

for t=1:N,

    % noises
    v1(:,t)=mvnrnd(zeros(1,n),Bv1*Bv1')';
    v2(t)=sqrt(V2)*randn;

    % system
    x(:,t+1)=A*x(:,t)+B*u(t)+v1(:,t);
    y(t)=C*x(:,t)+D*u(t)+v2(t);
end
Cov_v1=cov(v1'), V1 % To verify that: Var(v1) approximates V1
Cov_v2=cov(v2'), V2 % To verify that: Var(v2) approximates V2

% Step 6: plot of LTI system states and output

T=1:N;
for k=1:n,
    figure, plot(T,x(k,1:N),'g'), title(['State x_',num2str(k),'(t)'])
end
figure, plot(T,y(1:N),'g'), title('Output y(t)')

%% Dynamic predictor K and filter F in standard form

% Step 7: dynamic predictor K initialization

x_h(:,1)=zeros(n,1);
P{1}=0.5*eye(n);

% Step 8: dynamic predictor K and filter F simulation

for t=1:N,

    % dynamic predictor K
    y_h(t)=C*x_h(:,t);
    e(t)=y(t)-y_h(t);
    K{t}=(A*P{t}*C'+V12)*inv(C*P{t}*C'+V2);
    x_h(:,t+1)=A*x_h(:,t)+B*u(t)+K{t}*e(t);
    P{t+1}=A*P{t}*A'+V1-K{t}*(C*P{t}*C'+V2)*K{t}';

    % dynamic filter F
    K0{t}=P{t}*C'*inv(C*P{t}*C'+V2);
    x_f(:,t)=x_h(:,t)+K0{t}*e(t);
    y_f(t)=C*x_f(:,t);
end
K_N=K{N}
K0_N=K0{N}

% Step 9: RMSE computation

for ind=1:length(N0_vector),
    N0=N0_vector(ind);
    for k=1:n,
        RMSE_x_h(k,ind)=norm(x(k,N0+1:N)-x_h(k,N0+1:N))/sqrt(N-N0);
        RMSE_x_f(k,ind)=norm(x(k,N0+1:N)-x_f(k,N0+1:N))/sqrt(N-N0);
    end
    RMSE_y_h(ind)=norm(y(N0+1:N)-y_h(N0+1:N))/sqrt(N-N0);
```

```
    RMSE_y_f(ind)=norm(y(N0+1:N)-y_f(N0+1:N))/sqrt(N-N0);
end
fprintf('\n   N0 = %d    N0 = %d   N0 = %d\n',N0_vector(1),N0_vector(2),N0_vector(3))
RMSE_x_h, RMSE_y_h, RMSE_x_f, RMSE_y_f

% Step 10: graphical comparison of the results

for k=1:n,
    figure, plot(T,x(k,1:N),'g', T,x_h(k,1:N),'r-.', T,x_f(k,1:N),'b--'),
    title(['State x_',num2str(k),'(t)']), legend('System S','Predictor K','Filter F')
end
figure, plot(T,y(1:N),'g', T,y_h(1:N),'r-.', T,y_f(1:N),'b--'),
title('Output y(t)'), legend('System S','Predictor K','Filter F')

%% Steady-state predictor Kinf and filter Finf in standard form

% Step 11: steady-state predictor Kinf initialization

x_h_ss(:,1)=zeros(n,1);

% Off-line computation of steady-state Kalman gain matrices
Sys1=ss(A,[B, eye(n)],C,[D, zeros(1,n)],1);
[Kalman_predictor,Kbar,Pbar,K0bar]=kalman(Sys1,V1,V2,0);
Kbar              % To verify that: Kbar  = K_N
K0bar             % To verify that: K0bar = K0_N
A_K0bar=A*K0bar   % To verify that: Kbar  = A*K0bar

% Step 12: steady-state predictor Kinf and filter Finf simulation

for t=1:N,

    % steady-state predictor Kinf
    y_h_ss(t)=C*x_h_ss(:,t);
    e_ss(t)=y(t)-y_h_ss(t);
    x_h_ss(:,t+1)=A*x_h_ss(:,t)+B*u(t)+Kbar*e_ss(t);

    % steady-state filter Finf
    x_f_ss(:,t)=x_h_ss(:,t)+K0bar*e_ss(t);
    y_f_ss(t)=C*x_f_ss(:,t);
end

% Step 13: RMSE computation

for ind=1:length(N0_vector),
    N0=N0_vector(ind);
    for k=1:n,
        RMSE_x_h_ss(k,ind)=norm(x(k,N0+1:N)-x_h_ss(k,N0+1:N))/sqrt(N-N0);
        RMSE_x_f_ss(k,ind)=norm(x(k,N0+1:N)-x_f_ss(k,N0+1:N))/sqrt(N-N0);
    end
    RMSE_y_h_ss(ind)=norm(y(N0+1:N)-y_h_ss(N0+1:N))/sqrt(N-N0);
    RMSE_y_f_ss(ind)=norm(y(N0+1:N)-y_f_ss(N0+1:N))/sqrt(N-N0);
end
fprintf('\n   N0 = %d    N0 = %d   N0 = %d\n',N0_vector(1),N0_vector(2),N0_vector(3))
RMSE_x_h_ss, RMSE_y_h_ss, RMSE_x_f_ss, RMSE_y_f_ss

% Step 14: graphical comparison of the results

for k=1:n,
    figure, plot(T,x(k,1:N),'g', T,x_h_ss(k,1:N),'r-.', T,x_f_ss(k,1:N),'b--'),
    title(['State x_',num2str(k),'(t)']),
    legend('System S','Predictor K^\infty','Filter F^\infty')
end
figure, plot(T,y(1:N),'g', T,y_h_ss(1:N),'r-.', T,y_f_ss(1:N),'b--'),
```

```
title('Output y(t)'), legend('System S','Predictor K^\infty','Filter F^\infty')

%% (Optional) Dynamic predictor Kpc in predictor-corrector form

% Step 15: dynamic predictor Kpc initialization

x_h_pc(:,1)=zeros(n,1);
P{1}=0.5*eye(n);

% Step 16: dynamic predictor Kpc simulation

for t=1:N,
    K0{t}=P{t}*C'*inv(C*P{t}*C'+V2);
    P0{t}=(eye(n)-K0{t}*C)*P{t};
    P{t+1}=A*P0{t}*A'+V1;
    y_h_pc(t)=C*x_h_pc(:,t);
    e_pc(t)=y(t)-y_h_pc(t);
    x_f_pc(:,t)=x_h_pc(:,t)+K0{t}*e_pc(t);
    x_h_pc(:,t+1)=A*x_f_pc(:,t)+B*u(t);
end
norm(x_h-x_h_pc,inf) % To verify that: Kpc = K
norm(x_f-x_f_pc,inf) % To verify that: Fpc = F
```