ME C231B Assignment: Dynamics/Estimation Exam

Please read below, and do not turn in solutions unless you agree to the paragraph and sign below:

"This is to certify that I am receiving my copy of the ME C231B/EE C220C midterm exam. My submission of my solutions will act as a confirmation of my understanding that I may not discuss the contents of this exam with anyone until the complete exam period has ended. The exam period is April 5th, 6:00 PM - April 6th, 11:59 PM"

Student Name: Jun Zeng
SSID: 3033137317
Signature:

Midterm is due Friday, April 6th, 11:59PM.

- Submit a single zip file containing one main document and supporting .m files via bCourses.
- All exam solutions should be typed or neatly handwritten. Neat hand-drawn sketches are allowed within the document, as are neat handwritten equations.
- You will need to use Matlab (or other computational environments) in determining some of your solutions. Please make sure you can access these computational tools before downloading the exam.
- The copier on 6th floor of Etcheverry is available 24/7, and can be used without a login to email scans to any `berkeley.edu` address.

$= (g\sin\theta + a\cos\theta)\vec{x}_B + (g\cos\theta - a\sin\theta)\vec{z}_B$

Answer:

Since in the inertial frame

$\vec{a}^{sen}_{CoM} = g\vec{z}_I + a\vec{x}_I$

and.

$\vec{z}_I = \cos\theta\,\vec{z}_B + \sin\theta\,\vec{x}_B$

$\vec{x}_I = \cos\theta\,\vec{x}_B - \sin\theta\,\vec{z}_B$

thus

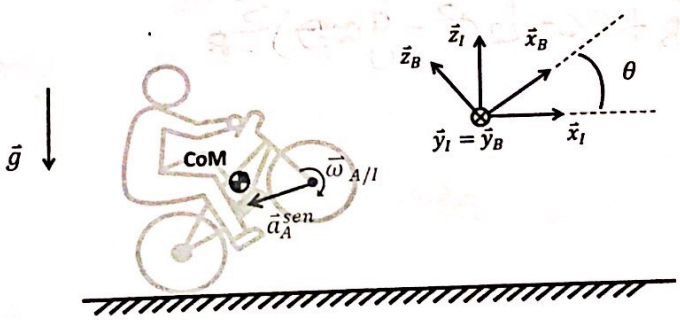
$\vec{a}^{sen}_{CoM} = g(\cos\theta\vec{z}_B + \sin\theta\vec{x}_B) + a(\cos\theta\vec{x}_B - \sin\theta\vec{z}_B)$

which means

$\alpha = g\sin\theta + a\cos\theta$

$\gamma = g\cos\theta - a\sin\theta$

(b) At another point in time the rider starts bringing the front axle towards the ground. The accelerometer in A measures $\vec{a}^{sen}_A = a\vec{x}_s + c\vec{z}_s$ (where a and c are given constants) and the gyro (mounted in A) measures a *constant* angular velocity $\vec{w}_{A/I} = \sigma\vec{y}_s$. What is the bike CoM acceleration?
Please answer the question in the form of $\vec{a}_{CoM} = \alpha\vec{x}_B + \gamma\vec{z}_B$. Note I asked the "CoM acceleration", not what the accelerometer mounted at the CoM measures.



(NOTE: As discussed in class the angular velocity $\vec{w}_{A/I}$ is the same as $\vec{w}_{CoM/I}$. Do not confuse the angular velocity $\vec{w}_{A/I}$ with the wheel angular speed. There are no wheels in this exercise and all IMUs are mounted on the bike frame.). *Answer:*

Solution:

$\vec{a}_{COM} = \vec{a}_A + \vec{a}_{COM/A}$ ($\vec{a}_{COM/A}$ is the relative acceleration)

where $\vec{a}_A = \vec{a}_A^{Sen} - g\vec{Z}_I$

$\vec{a}_{COM/A} = \omega^2\sqrt{L_1^2+L_2^2}\,\vec{e}_{COM\to A}$

and $\vec{e}_{COM\to A}$ is the unitare vector from center of mass to A.

$\vec{e}_{COM\to A} = (L_1\vec{X}_S - L_2\vec{Z}_S)/\sqrt{L_1^2+L_2^2}$

thus $\vec{a}_{COM} = [(a\vec{X}_S + c\vec{Z}_S) - g\vec{Z}_I] + \frac{L_1\vec{X}_S - L_2\vec{Z}_S}{\sqrt{L_1^2+L_2^2}}\cdot\sqrt{L_1^2+L_2^2}\,\omega^2$

$= (a+L_1\omega^2)\vec{X}_S + (c-L_2\omega^2)\vec{Z}_S - g\vec{Z}_I$

$= (a+L_1\omega^2)\vec{X}_B + (c-L_2\omega^2)\vec{Z}_B - g(\cos\theta\,\vec{Z}_B + \sin\theta\,\vec{X}_B)$

$= (a+L_1\omega^2 - g\sin\theta)\vec{X}_B + (c-L_2\omega^2 - g\cos\theta)\vec{Z}_B$

2. Quadcopter failure. 20pts

This problem uses the same notation, the same motor numbering and the same symbols of the document "DroneDynamics.pdf".
A hovering quadcopter is initially stationary (hovering at zero longitudinal and angular speeds and at a given constant height from the ground). Motor number 3 fails, and suddenly produces zero force/torque. The vehicle parameters are given below (in MKS)

$$\begin{aligned} m &= 0.0680 \\ d &= 0.0624 \\ b &= 0.0107 \\ k &= 7.8264e-04 \\ I_B &= diag([0.0686e-3,\ \ 0.092e-3,\ \ 0.1366e-3]) \end{aligned} \tag{1}$$

(a) Find the instantaneous acceleration of the quadcopter $\dot{v}_x$, $\dot{v}_y$, $\dot{v}_z$

on page 8. we have $\begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = A \cdot \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}$

To evaluate $\dot{v}_x, \dot{v}_y, \dot{v}_z$, we focus on T.

since initially hovering $\omega_1=\omega_2=\omega_3=\omega_4=\omega$

so $\dot{v}_z = F_{change}^{ext}/m$

$= +b\omega^2/m$

where $4b\omega^2 = mg$

thus $\dot{v}_z = \frac{1}{4}g$

since hovering $Q_{B/W} = \cdot$

$[F_{ext}] = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - Q_{B/W}\begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}$

the components of F_{ext} in x, y are zero.

$\dot{v}_x = \dot{v}_y = 0$

(b) Find the instantaneous angular acceleration of the quadcopter $\dot{\omega}_x$, $\dot{\omega}_y$, $\dot{\omega}_z$

$\begin{bmatrix} T \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = A \begin{bmatrix} \omega^2 \\ \omega^2 \\ \omega^2 \\ \omega^2 \end{bmatrix}$ hovering

thus

$$\begin{cases} \tau_x' = \tau_x \\ \tau_y' = db\omega^2 + \tau_y \\ \tau_z' = -k\omega^2 + \tau_z \end{cases}$$

$\begin{bmatrix} T' \\ \tau_x' \\ \tau_y' \\ \tau_z' \end{bmatrix} = A \begin{bmatrix} \omega^2 \\ \omega^2 \\ 0 \\ \omega^2 \end{bmatrix}$ now

thus

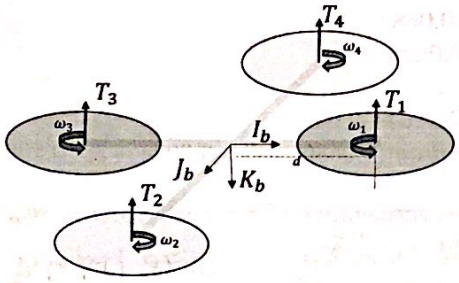
$$\begin{cases} \dot{\omega}_x = 0 \\ \dot{\omega}_y = db\omega^2/I_{B2} = \frac{db}{I_{B2}} \cdot \frac{mg}{4b} = \frac{mgd}{4I_{B2}} \\ \dot{\omega}_z = -k\omega^2/I_{B3} = \frac{-k}{I_{B3}} \cdot \frac{mg}{4b} = \frac{-kmg}{4I_{B3}b} \end{cases}$$

numerically

$\dot{\omega}_x = 0 \quad \dot{\omega}_y = 113.0\ rad/s^2 \quad \dot{\omega}_z = -892.1\ rad/s^2$

3. **Quadcopter Modeling. 10pts**

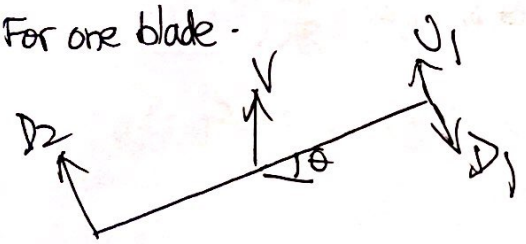
Consider the drone model studied in class and represented in the next figure.



- Explain why propellers 1 and 3 need to rotate in the opposite direction of propeller 2 and 4. Use the equations of motions derived in class to help your explanation.

Propellers 1 and 3 need to rotate in the opposite direction of propeller 2 and 4 in order to reduce the influence of "blade flapping" phenomena.

For one blade:



where the drone moves with a speed V.

the average force

$$\bar{F}_x = \frac{1}{2\pi}\int_0^{2\pi} F_x(\theta)\,d\theta \sim \omega v$$

$$\bar{F}_z = \frac{1}{2\pi}\int_0^{2\pi} (L_1+L_2) \sim \hat{L}_1\omega^2 + \hat{L}_3\omega^2$$

thus if four propellers rotate in the same directions, the system will have an additional force proportional to V in the direction x, y

when 1,3 rotate in the opposite direction of 2,4

$\omega_{1,3} = -\omega_{2,4}$

the average force from "blade flapping" in direction x and y will be compensated.

Moreover the torque in the direction z will be not zero when hovering, and A will become

$$A = \begin{bmatrix} -b & -b & -b & -b \\ 0 & -d\cdot b & 0 & d\cdot b \\ d\cdot b & 0 & -d\cdot b & 0 \\ K & +K & K & +K \end{bmatrix}$$

Where τ_z will be always positive.

- On our drone (and any quadrotor) we have two type of blades shown in the next figure. The design of blade B is symmetric to blade A. Explain why we need two different shapes. You

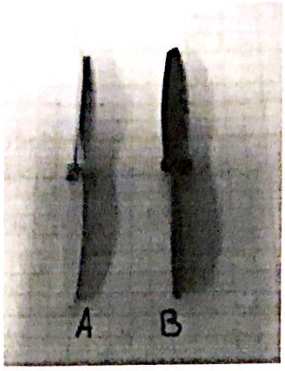


Figure 1: Blade types.

can use sketches and equations to help your explanation.

Since we have seen in the previous question, propellers A, C rotate in the opposite direction to B and D. if they use same blade types. the support force B and D. will be point to ground, in that case.

$$A = \begin{bmatrix} -b & b & -b & b \\ 0 & db & 0 & -db \\ db & 0 & -db & 0 \\ k & -k & k & -k \end{bmatrix}$$

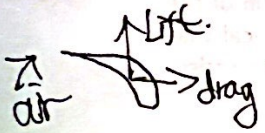
and in this case when $w_1 = w_2 = w_3 = w_4 = w$

Fext = 0 !! (we should avoid this)

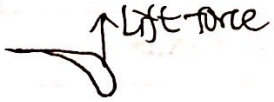
Precisely, we can look into the aerodynamic structure, in the lecture.

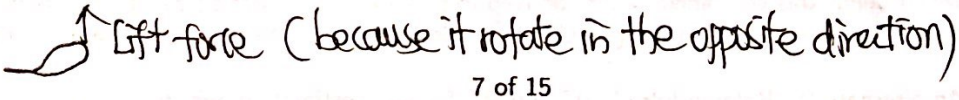
$L \sim w_j^2$

$D \sim w_j^2$



For our model

For A, C 

For B, D 

In the summary. because B, D rotate in the opposite direction, B, D need different shapes of blades when compared to A, C.

Kalman Filter

(a) $x_{K+1} = AX_K + EW_K$

$Z_{K+1} = A_K Z_K + B_K (C X_K + F W_K)$

$= A_K Z_K + B_K C X_K + B_K F W_K$

$e_K = -(X_K^{est} - X_K)$

$= -C_K Z_K - D_K Y_K + X_K$

$= -C_K Z_K - D_K (C X_K + F W_K) + X_K$

$= -C_K Z_K - (D_K C - I) X_K - D_K F W_K$

$$\begin{bmatrix} x_{K+1} \\ Z_{K+1} \\ e_K \end{bmatrix} = \begin{bmatrix} A & 0 & E \\ B_K C & A_K & B_K F \\ -D_K C + I & -C_K & -D_K F \end{bmatrix} \begin{bmatrix} X_K \\ Z_K \\ W_K \end{bmatrix}$$

(b) refer to combineProcessEstimator.m

(c) actually $A_K = \begin{bmatrix} 0 & I & & 0 \\ 0 & & \ddots & \\ & & & I_{ny} \\ 0 & & & 0 \end{bmatrix}$ $B_K = \begin{bmatrix} 0 \\ \vdots \\ I_{ny} \end{bmatrix}$

thus $\begin{cases} Z_{K+1}(i) = Z_K(i+1) \\ Z_{K+1}(N) = Y_K \end{cases} \Rightarrow Z_K(N) = Y_{K-1}$

recursively

$Z_K(1) = Z_{K-1}(2) = \cdots Z_{K-N}(N) = Y_{K-N}$

thus $Z_K = \begin{bmatrix} Y_{K-N} \\ \vdots \\ Y_{K-1} \end{bmatrix}$

Table of Contents

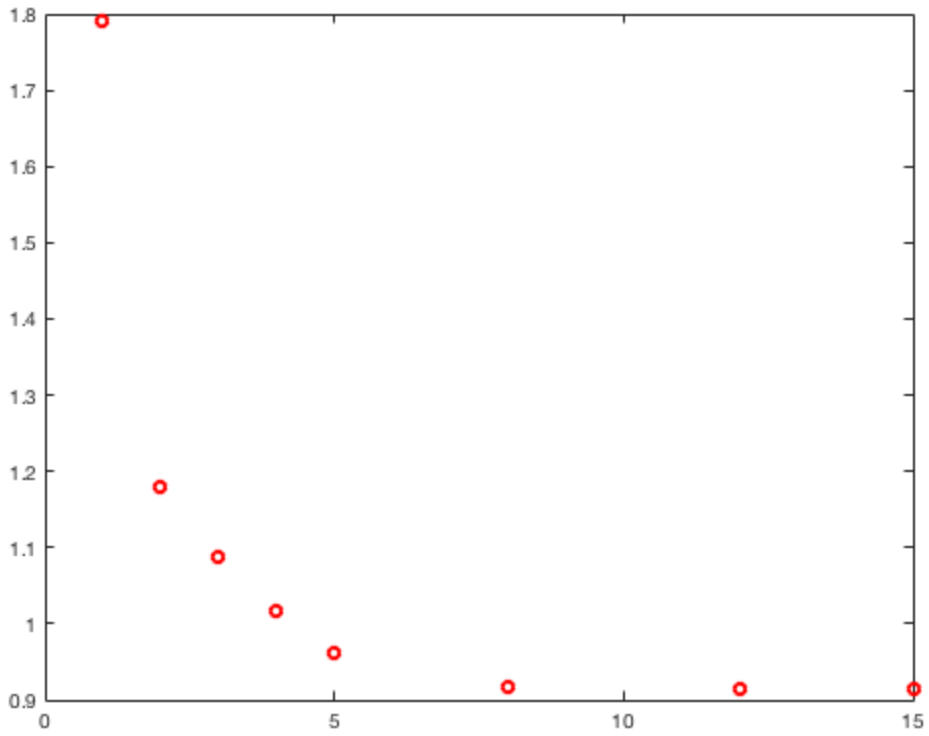
Midterm (Jun Zeng)

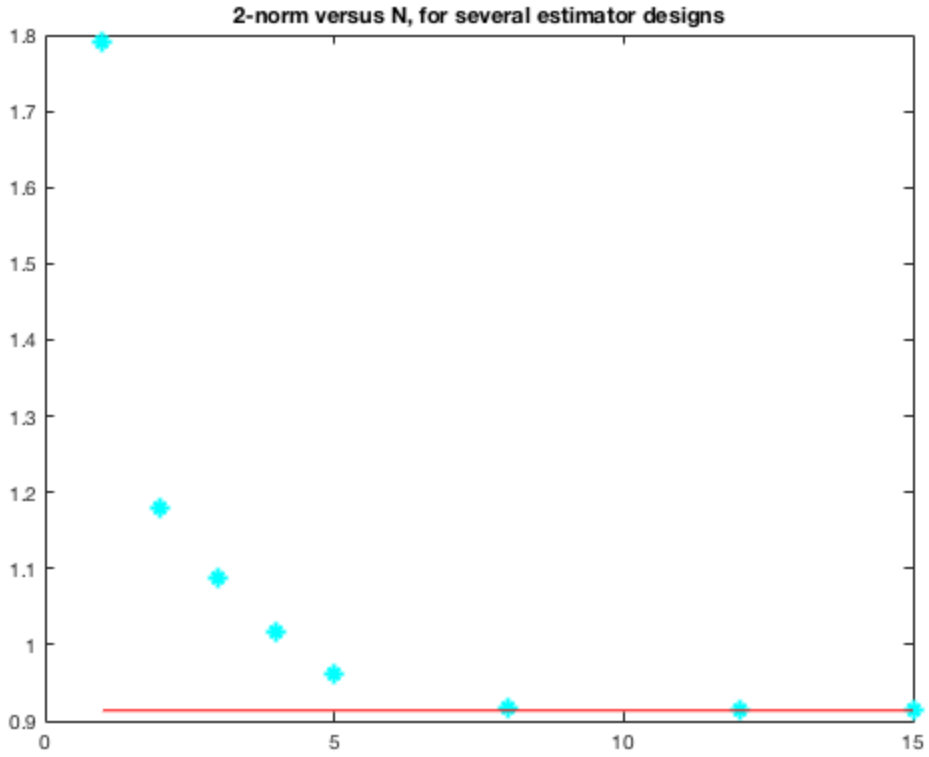
```
close all
options = optimoptions(@fminunc,'Display','none');
```

4.g

```
load('midtermModel1.mat')
NList = [1, 2, 3, 4, 5, 8, 12, 15];
cost1 = zeros(1,length(NList));
for i = 1:length(NList)
    N = NList(i);
    f = @(Lmat) evaluateCost(A,E,C,F,Lmat);
    LmatInit = zeros(nX,(N+1)*nY);
    [Lopt, optCost] = fminunc(f, LmatInit, options);
    [AK,BK,CK,DK] = createKfromLmat(Lopt,nX,nY,N);
    G1 = combineProcessEstimator(A,E,C,F,AK,BK,CK,DK);
    cost1(i) = norm(G1,2);
end
plot(NList,cost1,'ro');

W = eye(size(E, 2));
TS = -1;
[K,L,H,G,nIter,estErrVar] = formSteadyStateKF(A,E,C,F,W,TS);
fprintf('Estimation xkk')
K.OutputGroup.xkk

[K,L,H,G,nIter,estErrVar] = formSteadyStateKF(A,E,C,F,W,TS);
GsysK = combineProcessEstimator(A,E,C,F,A-L*C,L,eye(nX)-H*C, H);
costKF = norm(GsysK, 2);

% calculate covar
covarKF = covar(GsysK, eye(nW))
Gsys15 = combineProcessEstimator(A,E,C,F,AK,BK,CK,DK);
covarL15 = covar(Gsys15, eye(nW))

cost2 = zeros(length(NList),1);
for i = 1:length(NList)
    cost2(i) = costKF;
```

```
end
figure
plot(NList, cost1, 'c*')
hold on
plot(NList, cost2,'r')
title('2-norm versus N, for several estimator designs')

Estimation xkk
ans =

     4     5     6


covarKF =

    0.5291    0.0546   -0.0660
    0.0546    0.1985    0.0711
   -0.0660    0.0711    0.1074


covarL15 =

    0.5291    0.0546   -0.0660
    0.0546    0.1985    0.0711
   -0.0660    0.0711    0.1074
```





4.h

variables that should exist: process model (A, E, C, F) consistent dimensions N, positive integer, dictates complexity of estimator nX and nY, consistent with (A,C) Create the function handle for the cost function. The decision variable is the matix Lmat = @(Lmat) evaluateCost(A,E,C,F,Lmat);

```
load('testCaseSingleProcessModel.mat')
NList = [1, 2, 3, 4, 5, 8, 12, 15];
cost1 = zeros(1,length(NList));
costList2 = zeros(1,length(NList));
CovarList = zeros(1,length(NList));
for i = 1:length(NList)
    N = NList(i);
    f = @(Lmat) evaluateCost(A,E,C,F,Lmat);
    LmatInit = zeros(nX,(N+1)*nY);
    [Lopt, optCost] = fminunc(f, LmatInit, options);
    [AK,BK,CK,DK] = createKfromLmat(Lopt,nX,nY,N);
    G1 = combineProcessEstimator(A,E,C,F,AK,BK,CK,DK);
    cost1(i) = norm(G1,2);
end

% Steady State Testing
N = 15;
TS = -1;
W = eye(size(E,2));
[K,L,H,G,nIter,estErrVar] = formSteadyStateKF(A,E,C,F,W,TS);
```

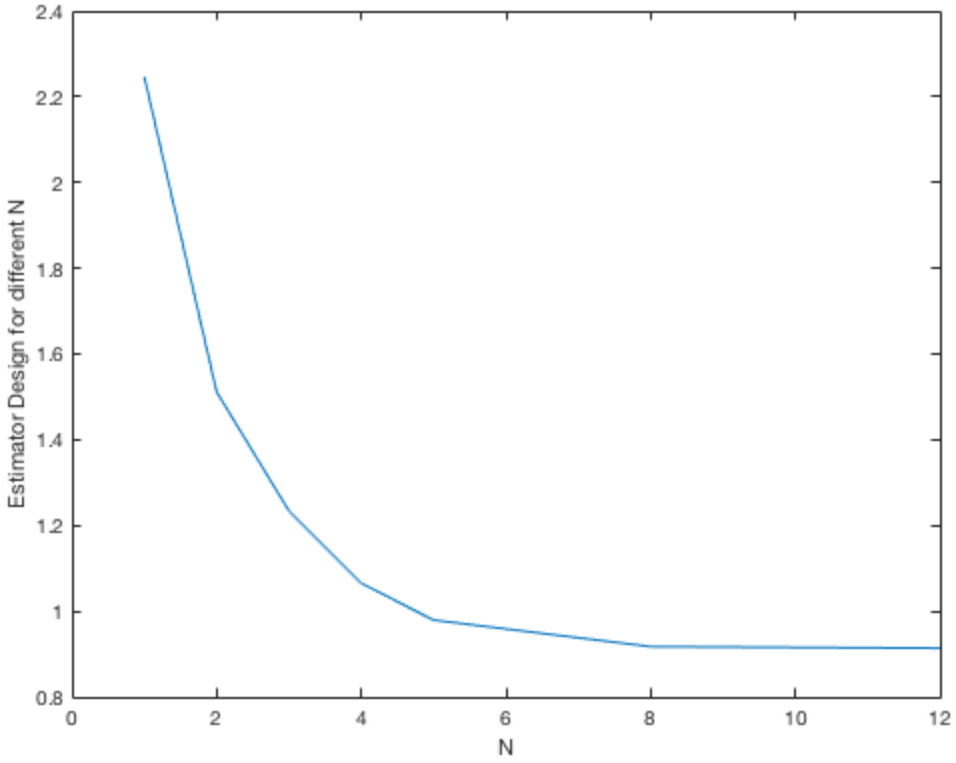
```
disp(estErrVar)
```

```
    1.2676    0.6449   -0.1706
    0.6449    0.3326   -0.0825
   -0.1706   -0.0825    0.2383
```

4.j

Robust Design

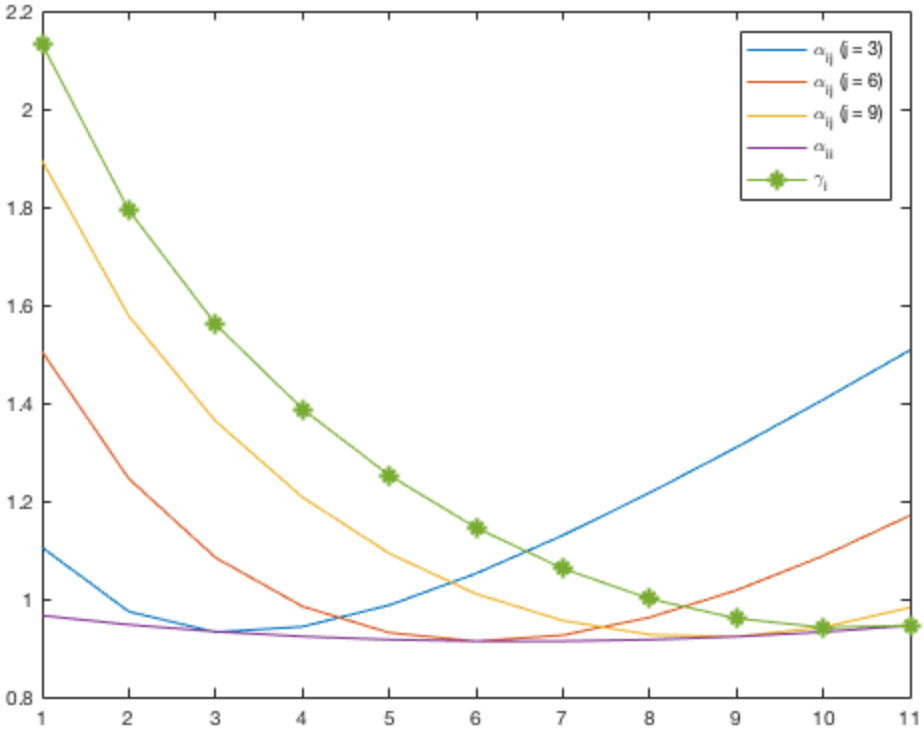
```
load('midtermMultipleModels.mat')
NList = [1,2,3,4,5,8,12];
cost1 = zeros(1,length(NList));
for i = 1:length(NList)
   N = NList(i);
   f = @(Lmat)
 evaluateWorstCostManyModels(A(:,:,i),E(:,:,i),C(:,:,i),F(:,:,i),Lmat);
   LmatInit = zeros(nX,(N+1)*nY);
   [Lopt, optCost] = fminunc(f, LmatInit, options);
   [AK,BK,CK,DK] = createKfromLmat(Lopt,nX,nY,N);
   G =
 combineProcessEstimator(A(:,:,i),E(:,:,i),C(:,:,i),F(:,:,i),AK,BK,CK,DK);
   cost1(i) = norm(G,2);
end
figure
plot(NList,cost1);
xlabel('N')
ylabel('Estimator Design for different N')
```



Robustness of the estimator

```
figure
for i = [3,6,9]
    Ai = A(:,:,i);
    Ei = E(:,:,i);
    Ci = C(:,:,i);
    Fi = F(:,:,i);
    W = eye(2);
    [K,L,H,G,nIter,estErrVar] = formSteadyStateKF(Ai,Ei,Ci,Fi,W,-1);
    cost1 = [];
    for j = 1:11
        Aj = A(:,:,j);
        Ej = E(:,:,j);
        Cj = C(:,:,j);
        Fj = F(:,:,j);
        M = [zeros(nX),eye(nX),zeros(nX,nW)];
        ICA = [Aj zeros(size(Aj,1),size(K.A,2));K.B*Cj K.A];
        ICB = [Ej;K.B*Fj];
        ICC = [eye(size(M,1))-M*K.D*Cj -M*K.C];
        ICD = -M*K.D*Fj;
        IC = ss(ICA,ICB,ICC,ICD,-1);
        cost1 = [cost1 norm(IC,2)];
    end
    plot(cost1);
```

```
    hold on
end

N = 12;
cost2 = zeros(11,1);
for i = 1:11
   f = @(Lmat)
 evaluateWorstCostManyModels(A(:,:,i),E(:,:,i),C(:,:,i),F(:,:,i),Lmat);
   LmatInit = zeros(nX,(N+1)*nY);
   [Lopt, optCost] = fminunc(f, LmatInit,options);
   [AK,BK,CK,DK] = createKfromLmat(Lopt,nX,nY,N);
   G =
 combineProcessEstimator(A(:,:,i),E(:,:,i),C(:,:,i),F(:,:,i),AK,BK,CK,DK);
   cost2(i) = norm(G,2);
end
plot(1:11,cost2);

N = 12;
costgamma = zeros(11,1);
for i = 1:size(A,3)
    [AK,BK,CK,DK] = createKfromLmat(Lopt,nX,nY,N);
    Gsys =
 combineProcessEstimator(A(:,:,i),E(:,:,i),C(:,:,i),F(:,:,i),AK,BK,CK,DK);
    cost = norm(Gsys, 2);
    costgamma(i) = cost;
end
plot(1:size(A,3), costgamma, '-*')
legend('\alpha_{ij} (j = 3)','\alpha_{ij} (j = 6)','\alpha_{ij} (j =
 9)','\alpha_{ii}','\gamma_i')
```



Comments

At first, we can see that the time variant kalman filter is better than the fixed time kalman filter and alternate one. Let's look at \alpha_{ij}, for a fixed value of j, the covariance of error in the conbination of jth kalman filter with i process model will reach the minimal value when i = j. We also plot \alpha_{ii} to reveal this comparaison.

Published with MATLAB® R2017b