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:..
SSID:..
Signature:..

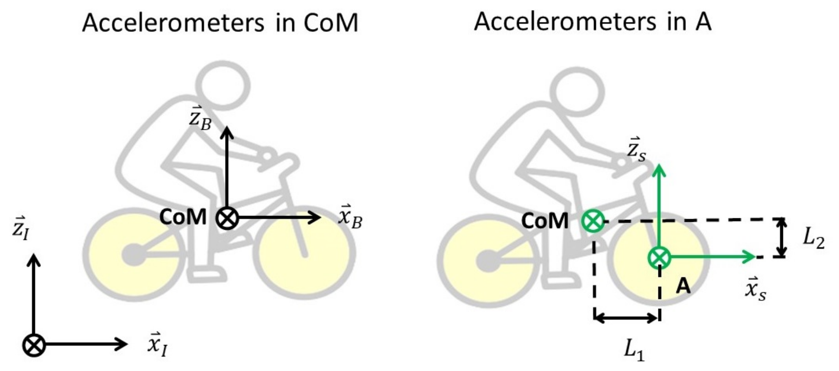
Midterm is due Friday, April 6th, 11:59PM.

- 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.

1. **Accelerometer readings. 20pts**
   We model a bicycle and its rider as a single 2D rigid body (we neglect the wheels dynamics). We introduce three coordinate systems:

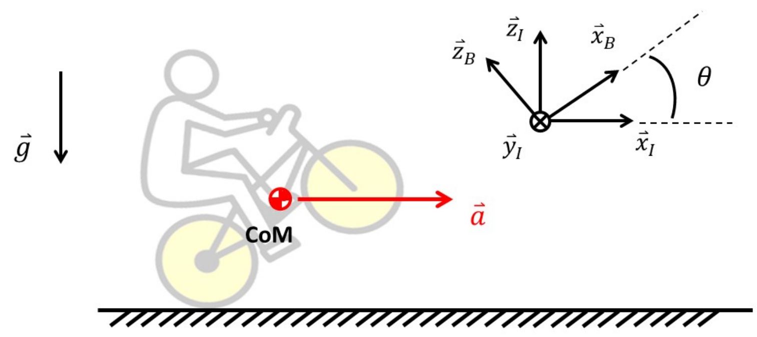
   1. An inertial frame fixed on the road and denoted as $[\vec{x}_I, \vec{y}_I, \vec{z}_I]$;
   2. A body-fixed frame located at the center of mass and denoted as $[\vec{x}_B, \vec{y}_B, \vec{z}_B]$.
   3. Another body-fixed frame located at the front wheel axle (point A) and with axes parallel to $[\vec{x}_B, \vec{y}_B, \vec{z}_B]$ and denoted as $[\vec{x}_S, \vec{y}_S, \vec{z}_S]$.

   Assume you have two IMUs (each one composed of accelerometer and a gyro). The first IMU is mounted at the center of mass (point CoM) and the second IMU is mounted the front wheel axle (point A) as shown in the next figure.



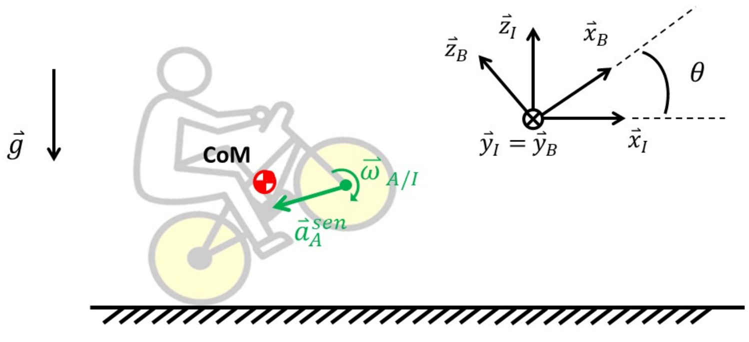
   Please answer the following questions.

   (a) The rider does a wheelie with the bike and at some point in time it moves forward with constant acceleration $\vec{a}$ of the CoM (with respect to the inertial frame) with direction shown in the figure below and magnitude a (i.e. $\vec{a} = a\vec{x}_I$). What value does the accelerometer mounted in CoM measures?
   Please answer the question in the form of $\vec{a}^{sen}_{CoM} = \alpha\vec{x}_B + \gamma\vec{z}_B$.



Answer:

(b) At another point in time the rider starts bringing the front axle towards the ground. The accelerometer in A measures $\vec{a}_A^{sen} = a\vec{x}_s + c\vec{z}_s$ (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:*

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)

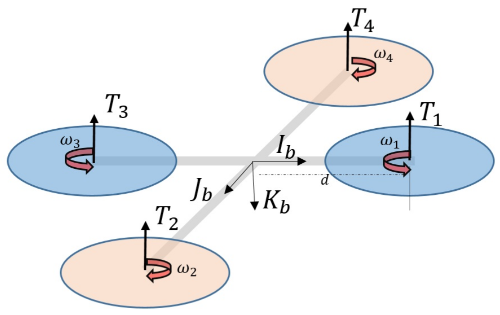
$$\begin{aligned} m &= 0.0680 \\ d &= 0.0624 \\ b &= 0.0107 \\ k &= 7.8264e-04 \\ I_B &= diag([0.0686e-3, \quad 0.092e-3, \quad 0.1366e-3]) \end{aligned} \tag{1}$$

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

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

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.

- 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

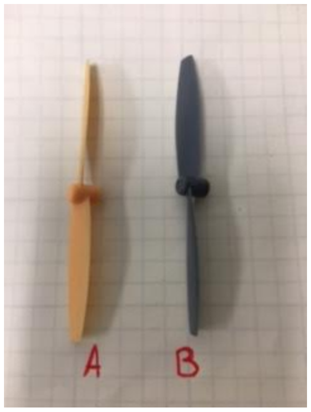


Figure 1: Blade types.

can use sketches and equations to help your explanation.

4. **Kalman filtering. 50pts**

(a) Concisely, the state equations for any time-invariant, linear estimator K which measures y and produces an estimate x_k^{est}, and has its own internal state z, are written as

$$\begin{aligned} z_{k+1} &= A_K z_k + B_K y_k \\ x_k^{\text{est}} &= C_K z_k + D_K y_k \end{aligned} \tag{2}$$

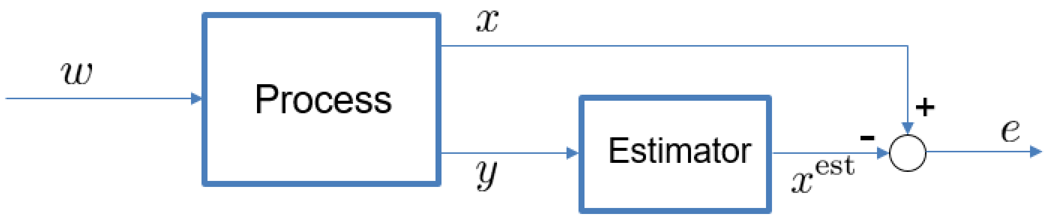
where y is the measured output of a process model, driven by a random process w,

$$\begin{aligned} x_{k+1} &= Ax_k + Ew_k \\ y_k &= Cx_k + Fw_k \end{aligned} \tag{3}$$

whose state, x is to be estimated. Define the estimation error $e_k := x_k - x_k^{\text{est}}$. Combine the time-invariant process model, (3), with the linear estimator (2) to obtain the equations which govern the relationship between w and e.

$$\begin{bmatrix} x_{k+1} \\ z_{k+1} \\ e_k \end{bmatrix} = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} x_k \\ z_k \\ w_k \end{bmatrix}$$

This is depicted in the figure below



So, your task is to determine the entries of the matrix above (all marked with $\cdots$). These will be expressions involving $A, E, C, F, A_K, B_K, C_K, D_K$ as well as identity and zero matrices. For later reference, notate this combined system (input w, output e) as G, or more specifically $G(A, E, C, F, A_K, B_K, C_K, D_K)$, illustrating its dependence on all of the matrices, comprising the Process and Estimator. (Note that this task is a simpler version of the problem you solved in *Kalman Filtering (Part 3)*, problem #5). **Remark**: It is critical that you get this problem (and the code in the next part) correct. The computational exercises we carry out later rely on this code being correct.

(b) Write a function, `combineProcessEstimator.m`, with function declaration-line

begin code
```
function Gsys = combineProcessEstimator(A,E,C,F,AK,BK,CK,DK)
```
end code

which builds this combined system derived in part (4a) (as a Matlab `ss` object). Make sure to set the sample-time to -1 (which means discrete-time, with unspecified sampled time).

(c) An alternate (to Kalman Filtering) approach for state estimation, which will not be optimal (but has some other advantages) is mentioned early in the slides. Here we investigate this further. Matrices $L_0, L_1, \ldots, L_N$ are fixed, and for each k, the state estimate of x_k is simply a combination of the current measurement y_k along with the previous N measurements, $y_{k-1}, y_{k-2}, \ldots, y_{k-N}$. Mathematically, this is

$$x_k^{\text{est}} = L_0 y_k + L_1 y_{k-1} + L_2 y_{k-2} + \ldots + L_N y_{k-N} \tag{4}$$

Each $L_i \in \mathbb{R}^{n_x \times n_y}$ are chosen by some optimization criteria, once the process model, (3), is given.

Let K denote the system mapping input y to output x^{est} through the formula

$$x_k^{\text{est}} = L_0 y_k + L_1 y_{k-1} + L_2 y_{k-2} + \ldots + L_N y_{k-N}$$

A suitable state-space realization for K is

$$A_K = \begin{bmatrix} 0_{n_y} & I_{n_y} & 0_{n_y} & 0_{n_y} & \cdots & 0_{n_y} & 0_{n_y} \\ 0_{n_y} & 0_{n_y} & I_{n_y} & 0_{n_y} & \cdots & 0_{n_y} & 0_{n_y} \\ 0_{n_y} & 0_{n_y} & 0_{n_y} & I_{n_y} & \cdots & 0_{n_y} & 0_{n_y} \\ 0_{n_y} & 0_{n_y} & 0_{n_y} & 0_{n_y} & \cdots & 0_{n_y} & 0_{n_y} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0_{n_y} & 0_{n_y} & 0_{n_y} & 0_{n_y} & \cdots & 0_{n_y} & I_{n_y} \\ 0_{n_y} & 0_{n_y} & 0_{n_y} & 0_{n_y} & \cdots & 0_{n_y} & 0_{n_y} \end{bmatrix}, \qquad B_K = \begin{bmatrix} 0_{n_y} \\ 0_{n_y} \\ 0_{n_y} \\ 0_{n_y} \\ \vdots \\ 0_{n_y} \\ I_{n_y} \end{bmatrix} \tag{5}$$

and

$$C_K = \begin{bmatrix} L_N & L_{N-1} & L_{N-2} & L_{N-3} & \cdots & L_2 & L_1 \end{bmatrix}, \qquad D_K = L_0 \tag{6}$$

where the dimensions of the matrices are

$$A_K \in \mathbb{R}^{n_y \cdot N \times n_y \cdot N}, \qquad B_K \in \mathbb{R}^{n_y \cdot N \times n_y}, \qquad C_K \in \mathbb{R}^{n_x \times n_y \cdot N}, \qquad D_K \in \mathbb{R}^{n_x \times n_y}$$

Task to complete in this part: Use the notation y for input to K and z for state of K, so that K is implemented as

$$\begin{aligned} z_{k+1} &= A_K z_k + B_K y_k \\ x_k^{\text{est}} &= C_K z_k + D_K y_k \end{aligned} \tag{7}$$

Show that for $k \geq N$, the structure of A_K and B_K in (5) imply that the state z_k is

$$z_k = \begin{bmatrix} y_{k-N} \\ y_{k-(N-1)} \\ \vdots \\ y_{k-2} \\ y_{k-1} \end{bmatrix},$$

regardless of initial condition. In other words, the state z_k merely holds the previous N values of the input.

(d) Let L_{mat} denote the concatenation of the L_i matrices, compatible with their appearance in C_K and D_K of the state-space realization of the estimator described in equation (6) in part (4c)

$$L_{\text{mat}} := \begin{bmatrix} L_N & L_{N-1} & L_{N-2} & L_{N-3} & \cdots & L_2 & L_1 & L_0 \end{bmatrix} \in \mathbb{R}^{n_x \times (N+1)n_y}$$

Write a function, called `createKfromLmat.m`, with function declaration shown below

```
function [AK,BK,CK,DK] = createKfromLmat(Lmat,nX,nY,N)
% verify that size of Lmat is [nX (N+1)*nY]
% extract CK matrix from Lmat
% extract DK matrix from Lmat
% create AK matrix using N, nY
% create BK matrix from N, nY
```

The purpose is to create the linear, discrete-time state-space matrices for K, described in part (4c), equations (5) and (6), given the matrix L_{mat}.

(e) (nothing to do - but read carefully) Recall from (*KalmanFiltering (Part 3)*, problem 3) this fact about time-invariant, stable linear systems: Suppose G is a stable, linear, time-invariant linear system with input u and output y. Suppose the input is zero-mean, unit-intensity white noise, namely for all k and all $j \neq k$

$$\mathbb{E}u_k = 0, \qquad \mathbb{E}u_k u_k^T = I_{n_u}, \qquad \mathbb{E}u_k u_j^T = 0_{n_u},$$

then

$$\left(\lim_{k\to\infty} y_k^T y_k\right)^{1/2} = \|G\|_2$$

This fact relates the frequency-response function of a stable system to its response to white-noise inputs.

(f) Refer back to parts (4a) and (4d). In part (4a), we use the notation $G(A, E, C, F, A_K, B_K, C_K, D_K)$ to denote the combined system for some arbitary estimator with state-space realization (A_K, B_K, C_K, D_K). In part (4d), we develop the mapping from L_{mat} to the matrices A_K, B_K, C_K, D_K for the specific type of estimator given by (4). Let's agree to modify the notation to represent this combined system as $G(A, E, C, F, K(L_{\text{mat}}))$, which accounts for the fact that the system K is determined from the elements in L_{mat}.

Since the 2-norm is mathematically equivalent to the steady-state output variance under the white-noise input, the estimator design *can be based on minimization* of the 2-norm, namely

$$\min_{L_{\text{mat}} \in \mathbb{R}^{n_x \times ((N+1)n_y}} \|G(A, E, C, F, K(L_{\text{mat}}))\|_2$$

Task: Write the following function which computes the $\|\cdot\|_2$ cost associated with an estimator (specified by the Lmat argument) and a process model. The function declaration line, along with some comments that outline the functionality of the code is below.

```
function cost = evaluateCost(A,E,C,F,Lmat)
% compute nX from A
% compute nY from C
% compute N from size of Lmat
% use createKfromLmat to obtain matrices AK,BK,CK,DK
% use combineProcessEstimator to create G
% use norm to obtain cost.
```

Remark: The cost function,

$$f(L_{\mathrm{mat}}) := \|G(A, E, C, F, K(L_{\mathrm{mat}}))\|_2$$

is a convex, differntiable function of L_{mat}. This is great news for optimization...

Note: With this cost-function specified, Matlab's unconstrained optimizer, `fminunc`, can be used to perform the optimization (and hence the design of the L matrices). A script-file to perform the design, assuming the process model is defined, could be of the form below

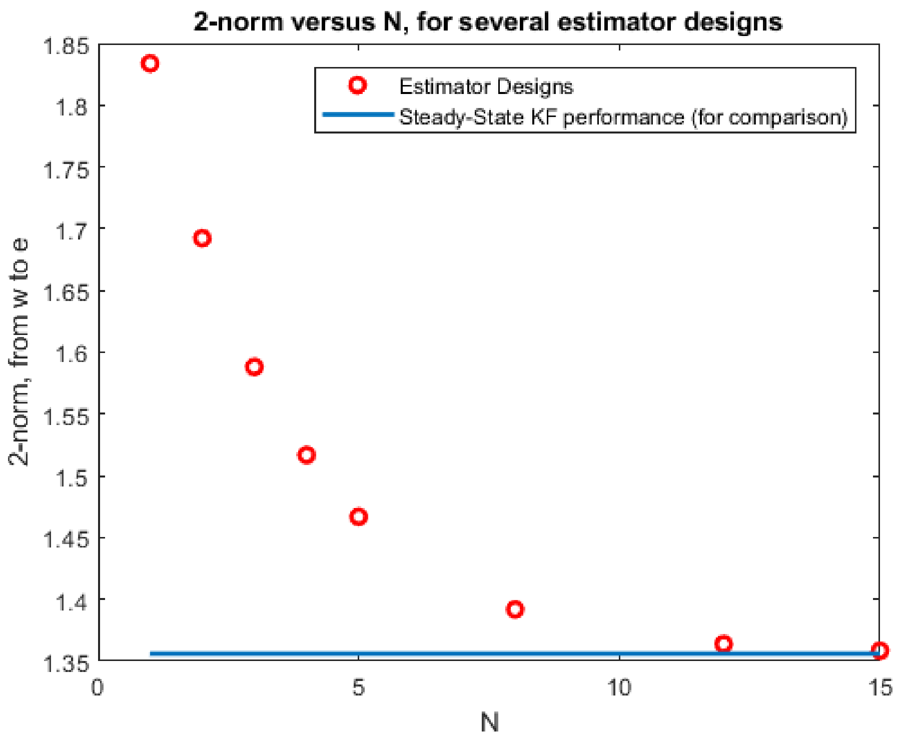
begin code

```
% 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
f = @(Lmat) evaluateCost(A,E,C,F,Lmat);
%
LmatInit = zeros(nX,(N+1)*nY);
% Call the solver, with the function handle for cost, and
% the initial value (just 0) for the decision variable matrix
[Lopt, optCost] = fminunc(f, LmatInit);
%
% Form the state-space model of the estimator
[AK,BK,CK,DK] = createKfromLmat(Lopt,nX,nY,N);
%
% Perform analysis on the estimator's performance...
```

end code

(g) Data for A, E, C, F is given in the file `midtermModel1.mat`. Use your code to solve for the estimator (4) which minimizes the $\|\cdot\|_2$ norm from w to e. Do this for several values of N, namely $N = 1, 2, 3, 4, 5, 8, 12, 15$. Plot the obtained norm, $\|G\|_2$ versus N, Since we are minimizing, the norm should decrease with increasing N. Also use the code `formSteadyStateKF.m` to compute the steady-state Kalman filter, and specifically the estimate $\hat{x}_{k|k}$ (please read the first few items in **Remarks/Hints** at the end of this problem). Let G_{KF} denote the combination of the process and Kalman filter, with input w and output $e_k := x_k - \hat{x}_{k|k}$. Compute $\|G_{KF}\|_2$ to compare with the results from the alternate form of the estimator. Use the `covar` command to compute the steady-state variance of the e_k, both for the Kalman filter, and for the design with $N = 15$. Arrange your results neatly and concisely.

(h) (**nothing to do here - for your own use**) In order to assist you in debugging, another example is provided in `testCaseSingleProcessModel.mat` The results (for the same tasks listed above in problem 4g) are shown below



```
estErrVarWithL15 =

    1.2679    0.6435   -0.1704
    0.6435    0.3391   -0.0833
   -0.1704   -0.0833    0.2384


estErrVarWithKF =

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

(i) **Robust Estimation:** An attractive feature of the optimization-based approach is the simple modification needed to handle uncertain process models. In the simplest case, suppose the process model is not a single model, but m separate models,

$$(A_1, E_1, C_1, F_1),\ (A_2, E_2, C_2, F_2),\ \dots\ , (A_m, E_m, C_m, F_m) \tag{8}$$

Here we assume that the dimensions are consistent across these m models. The *robust estimation* problem is to chose a single estimator $K(L)$ that works well for all the process models. One approach is to minimize the worst-cost associated with

$$\min_{L_{\text{mat}} \in \mathbb{R}^{n_x \times ((N+1)n_y}} \underbrace{\max_{1 \le i \le M} \|G(A_i, E_i, C_i, F_i, K(L_{\text{mat}}))\|_2}_{\text{worst-case cost across all process models}}$$

Write a command called `evaluateWorstCostManyModels.m`, with function declaration line

```
begin code
function cost = evaluateWorstCostManyModels(A,E,C,F,Lmat)
end code
```

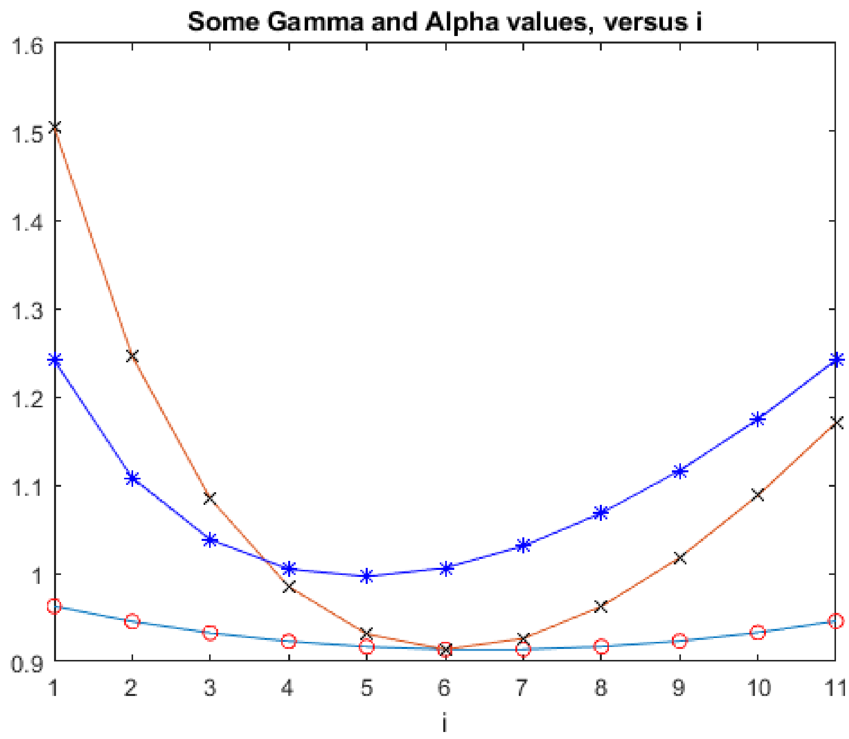
where input arguments A, E, C, F are 3-dimensional arrays, of dimension $n_x \times n_x \times m$, $n_x \times n_w \times m$, $n_y \times n_x \times m$, $n_y \times n_w \times m$ respectively, corresponding to the multple process models in equation (8). The argument `Lmat` corresponds to the estimator parameters, as before. The output argument `cost` is the worst-case cost associated with the estimator, namely

$$\max_{1 \le i \le m} \|G(A_i, E_i, C_i, F_i, K(L_{\text{mat}}))\|_2$$

Again, this cost function is a convex function of L_{mat}, but unfortunately, because of the $\max$ structure, it is not differentiable. Please read the **Remarks/Hints** for a bit more information (not necessary for the exam though).

(j) There is a data file, `midtermMultipleModels.mat` for a robust design, with 11 models ($m = 11$, represented with a variable `nModels` in the file). The (A, E, C, F) matrices are 3-dimensional, so `A(:,:,i)` is A_i, and i runs from 1 to 11. Careful examination reveals that the model is derived from the single-model example, `midtermModel1.mat`, *with one key difference*. The $(1, 2)$ element of A varies from -1 to 0, across the 11 models, whereas in part (4g) it was constant, with value equal to -0.5. In this example, the other matrices, E, C, F do not change along their 3rd dimension. **Task:**

- Carry out a robust design, using the data in `midtermMultipleModels.mat`, the cost function in `evaluateWorstCostManyModels.m`, and an appropriate call to `fminunc`. Do this for $N = 1, 2, 3, 4, 5, 8, 12$, recording the optimized worst-case cost for each design.
- Analyze more carefully the robustness of the estimator designed with $N = 12$. Read the **Hints/Remarks** section about assessing various combinations of estimators and process models (the γ and α arrays).
- **Make some appropriate plots, a few comments, and document your script file**.
- In order to check your work along the way, a relevant plot (I am not providing much information, but once you dig into this problem, the plot below might serve as a useful comparison) is shown below.



Remarks/Hints:

- When you use `formSteadyStateKF.m` to obtain Kalman filter for any given process model, be careful about the outputs. We are interested in comparing $\hat{x}_{k|k}$ to the result of the estimator in (4), since that estimator does use y_k in the estimate x_k^{est}. Be sure to use to correct outputs from the system returned by `formSteadyStateKF.m`. Remember that in addition to $\hat{x}_{k|k}$, the outputs include $\hat{x}_{k|k-1}$ and $\hat{w}_{k|k}$ which we are not interested in.
- The 5th input argument to `formSteadyStateKF.m` is the variance of w_k, assumed to be constant for all k. Since the $\|\cdot\|_2$ norm calculation relates to the output variance when the input is *zero-mean, unit-intensity white noise*, use I_{n_w} for the variance of w_k.
- In the robust problem, there are many possible design/calculations and comparisons to do...
  (a) For each choice of N, one robust estimator design. Call this design $K_{\mathrm{rob},N}$
  (b) One Kalman filter design at each process model (A_i, E_i, C_i, F_i). Use `formSteadyStateKF.m` and call this design KF_i .
  (c) The only performance metric is $\|G\|_2$, where G is the combination of one of the process models and one of the estimators. We are interested in 2 specific combinations
    - γ_i, for the robust estimator, $K_{\mathrm{rob},N}$ combined with the i'th process model, (A_i, E_i, C_i, F_i)
    - $\alpha_{i,j}$, for the i'th process model (A_i, E_i, C_i, F_i), combined with the j'th Kalman Filter, KF_j.

  By plotting $\alpha_{i,i}$ versus i, we see what is achievable at each fixed i, which acts as sort of baseline performance. By plotting $\alpha_{i,j}$ versus i (for fixed j) we see how the j'th Kalman filter performs across the entire set of process models (perhaps one of these Kalman Filters does well across the entire set of process model, or maybe not - perhaps the only way to achieve good robustness is to use the tools developed in part 4i). Finally, by plotting γ_i versus i, we see how the Robust filter performs across the entire set of process models. In terms of worst-case performance, this should be better than any one of the individual Kalman filter designs.... Is it?

Important Remark for later use (nothing to do on exam): The cost functions we have defined, for the single process model, and for the multiple-model case, **are both convex functions of the decision-variable** L_{mat}. Unfortunately, in the multiple-model case, the cost-function (since it involves a $\max$) is not differentiable. This nondifferentiabilty can cause general-purpose solvers, such as `fminunc` to prematurely stop. There are other more advanced solvers that can handle the non-differentiability, but that is not the focus of the problem. However, if you find yourself faced with a robust-estimator design problem, you may have to use alternative optimization methods to obtain a good solution with the approach ((4)) presented here.