

Midterm exam

This is a 24 hour take-home midterm. Please turn it in at Bytes Cafe in the Packard building, 24 hours after you pick it up.

- You may use any books, notes, or computer programs (*e.g.*, matlab), but you may not discuss the exam with others until Oct. 25, after everyone has taken the exam. The only exception is that you can ask the TAs or Stephen Boyd for clarification, by emailing to the staff email address `ee263-aut1011-staff@lists.stanford.edu`. We've tried pretty hard to make the exam unambiguous and clear, so we're unlikely to say much.
- Since you have 24 hours, we expect your solutions to be legible, neat, and clear. Do not hand in your rough notes, and please try to simplify your solutions as much as you can. We will deduct points from solutions that are technically correct, but much more complicated than they need to be.
- Please check your email a few times during the exam, just in case we need to send out a clarification or other announcement. It's unlikely we'll need to do this, but you never know.
- Attach the official exam cover page (available when you pick up or drop off the exam, or from the midterm info page) to your exam, and assemble your solutions to the problems in order, *i.e.*, problem 1, problem 2, . . . , problem 6. Start each solution on a new page.
- Please make a copy of your exam before handing it in. We have never lost one, but it might occur.
- When a problem involves some computation (say, using matlab), we do not want just the final answers. We want a clear discussion and justification of exactly what you did, the matlab source code that produces the result, and the final numerical result. Be sure to show us your verification that your computed solution satisfies whatever properties it is supposed to, at least up to numerical precision. For example, if you compute a vector x that is supposed to satisfy $Ax = b$ (say), show us the matlab code that checks this, and the result. (This might be done by the matlab code `norm(A*x-b)`; be sure to show us the result, which should be very small.) *We will not check your numerical solutions for you, in cases where there is more than one solution.*

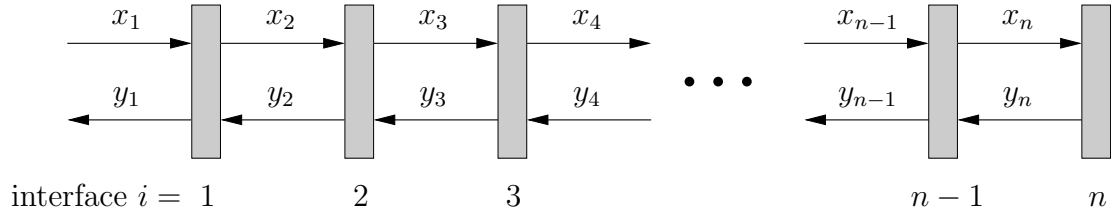
- In the portion of your solutions where you explain the mathematical approach, you *cannot* refer to matlab operators, such as the backslash operator. (You can, of course, refer to inverses of matrices, or any other standard mathematical construct.)
- Some of the problems are described in a practical setting, such as signal processing, finance, or biology. *You do not need to understand anything about the application area to solve these problems.* We've taken special care to make sure all the information and math needed to solve the problem is given in the problem description.
- Some of the problems require you to download and run a matlab file to generate the data needed. These files can be found at the URL

`http://www.stanford.edu/class/ee263/midterm_10_11_mfiles/FILENAME`

where you should substitute the particular filename (given in the problem) for `FILENAME`. *There are no links on the course web page pointing to these files, so you'll have to type in the whole URL yourself.*

- Please respect the honor code. Although we encourage you to work on homework assignments in small groups, *you cannot discuss the midterm with anyone*, with the exception of Stephen Boyd and the TAs, until everyone has taken it.

1. *Layered medium.* In this problem we consider a generic model for (incoherent) transmission in a layered medium. The medium is modeled as a set of n layers, separated by n dividing interfaces, shown as shaded rectangles in the figure below.



We let $x_i \in \mathbf{R}$ denote the right-traveling wave amplitude in layer i , and we let $y_i \in \mathbf{R}$ denote the left-traveling wave amplitude in layer i , for $i = 1, \dots, n$. The right-traveling wave in the first layer is called the incident wave, and the left-traveling wave in the first layer is called the reflected wave. The scattering coefficient for the medium is defined as the ratio $S = y_1/x_1$ (assuming $x_1 \neq 0$).

The right- and left-traveling waves on each side of an interface are related by transmission and reflection. The right-traveling wave of amplitude x_i contributes the amplitude $t_i x_i$ to x_{i+1} , where $t_i \in [0, 1]$ is the transmission coefficient of the i th interface. It also contributes the amplitude $r_i x_i$ to y_i , the left-traveling wave, where $r_i \in [0, 1]$ is the reflection coefficient of the i th interface. We will assume that the interfaces are symmetric, so the left-traveling wave with amplitude y_{i+1} contributes the wave amplitude $t_i y_{i+1}$ to x_i (via transmission) and wave amplitude $r_i y_{i+1}$ to x_{i+1} (via reflection). Thus we have

$$x_{i+1} = t_i x_i + r_i y_{i+1}, \quad y_i = r_i x_i + t_i y_{i+1}, \quad i = 1, 2, \dots, n-1.$$

We model the last interface as totally reflective, which means that $y_n = x_n$.

- (a) Explain how to find the scattering coefficient S , given the transmission and reflection coefficients for the first $n-1$ layers.
- (b) Carry out your method for a medium with $n = 20$ layers, and $t_i = 0.96$, $r_i = 0.02$ for $i = 1, \dots, n-1$. Plot the left- and right-traveling wave amplitudes x_i, y_i versus i , and report the value of S you find.
Hint: You may find the matlab function `diag(x,k)` useful.
- (c) *Fault location.* A fault in interface k results in a reversal: $t_k = 0.02$, $r_k = 0.96$, with all other interfaces having their nominal values $t_i = 0.96$, $r_i = 0.02$. You measure the scattering coefficient $S = S^{\text{fault}}$ with the fault (but you don't have access to the left- or right-traveling waves with the faulted interface). Explain how to find which interface is faulted. Carry out your method with $S^{\text{fault}} = 0.70$. You may assume that the last (fully reflective) interface is not faulty. Be sure to give the value of k that is most consistent with the measurement.

2. *State trajectory estimation.* We consider a discrete-time linear dynamical system

$$x(t+1) = Ax(t) + Bu(t) + w(t), \quad y(t) = Cx(t) + v(t), \quad t = 1, 2, \dots,$$

with state $x(t) \in \mathbf{R}^n$, input $u(t) \in \mathbf{R}^m$ and output $y(t) \in \mathbf{R}^p$. The signal $w(t)$ is called the process noise, and the signal $v(t)$ is the measurement noise. You know the matrices A , B , C , the inputs $u(t)$, $t = 1, 2, \dots, T-1$, and outputs $y(t)$, $t = 1, 2, \dots, T$. You do not know $x(t)$, $w(t)$, or $v(t)$. Your job is to estimate the state trajectory $x(t)$, for $t = 1, \dots, T$. We will denote your estimate of the state trajectory as $\hat{x}(t)$, $t = 1, \dots, T$. When we guess the state trajectory $\hat{x}(t)$, $t = 1, \dots, T$, we have two sets of residuals,

$$\hat{x}(t+1) - (A\hat{x}(t) + Bu(t)), \quad t = 1, \dots, T-1, \quad y(t) - C\hat{x}(t), \quad t = 1, \dots, T,$$

which correspond to (implicit) estimates of $w(t)$ and $v(t)$, respectively.

You will choose $\hat{x}(t)$ so as to minimize the overall objective

$$J = \sum_{t=1}^{T-1} \|\hat{x}(t+1) - (A\hat{x}(t) + Bu(t))\|^2 + \rho \sum_{t=1}^T \|y(t) - C\hat{x}(t)\|^2,$$

where $\rho > 0$ is a given parameter (related to our guess of the relative sizes of $w(t)$ and $v(t)$). The objective J is a weighted sum of norm squares of our two residuals.

- (a) Explain how to find the state trajectory estimate $\hat{x}(t)$, $t = 1, \dots, T$, using any concepts from the course. If one or more matrices must satisfy a rank condition for your method to work, say so.
- (b) Carry out your method from part (a) using `state_traj_estim_data.m`, which gives A , B , C , the dimensions n , m , p , the parameter ρ , and the time horizon T . The input and output trajectories are given as $m \times T$ and $p \times T$ matrices, respectively. (The t th column gives the vector at the t th period.)

Give the value of J corresponding to your estimate.

The mfile includes the true value of the state trajectory, $x(t)$, (of course you may not use it in forming your estimate $\hat{x}(t)$). Plot $x_1(t)$ (the true first state component) and $\hat{x}_1(t)$ (the estimated first state component) on the same plot.

Matlab hints:

- The matlab command `x = X(:)`, where X is an n by m matrix, stacks the columns of X into a vector of dimension nm . You may then recover X with the command `X = reshape(x,n,m)`.
- You might find the matlab function `blkdiag` useful.

3. *Reverse engineering a smoothing filter.* A smoothing filter takes an input vector $u \in \mathbf{R}^n$ and produces an output vector $y \in \mathbf{R}^n$. (We will assume that $n \geq 3$.) The output y is obtained as the minimizer of the objective

$$J = J^{\text{track}} + \lambda J^{\text{norm}} + \mu J^{\text{cont}} + \kappa J^{\text{smooth}},$$

where λ , μ , and κ are positive constants (weights), and

$$J^{\text{track}} = \sum_{i=1}^n (u_i - y_i)^2, \quad J^{\text{norm}} = \sum_{i=1}^n y_i^2$$

are the tracking error and norm-squared of y , respectively, and

$$J^{\text{cont}} = \sum_{i=2}^n (y_i - y_{i-1})^2, \quad J^{\text{smooth}} = \sum_{i=2}^{n-1} (y_{i+1} - 2y_i + y_{i-1})^2$$

are measures of the continuity and smoothness of y , respectively.

Here is the problem: You have access to one input-output pair, *i.e.*, an input u , and the associated output y . Your goal is to find the weights λ , μ , and κ . In other words, you will reverse engineer the smoothing filter, working from an input-output pair.

- (a) Explain how to find λ , μ , and κ . (You do not need to worry about ensuring that these are positive; you can assume this will occur automatically.)
- (b) Carry out your method on the data found in `rev_eng_smooth_data.m`. Give the values of the weights.

4. *Flux balance analysis in systems biology.* Flux balance analysis is based on a very simple model of the reactions going on in a cell, keeping track only of the gross conservation of various chemical species (metabolites) within the cell.

We focus on m metabolites in a cell, labeled M_1, \dots, M_m . There are n (reversible) reactions going on, labeled R_1, \dots, R_n , with reaction rates $v_1, \dots, v_n \in \mathbf{R}$. A positive value of v_i means the reaction proceeds in the given direction, while a negative value of v_i means the reaction proceeds in the reverse direction. Each reaction has a (known) stoichiometry, which tells us the rate of consumption and production of the metabolites per unit of reaction rate. The stoichiometry data is given by the *stoichiometry matrix* $S \in \mathbf{R}^{m \times n}$, defined as follows: S_{ij} is the rate of production of M_i due to unit reaction rate $v_j = 1$. Here we consider consumption of a metabolite as negative production; so $S_{ij} = -2$, for example, means that reaction R_j causes metabolite M_i to be consumed at a rate $2v_j$. If v_j is negative, then metabolite M_i is produced at the rate $2|v_j|$.

As an example, suppose reaction R_1 has the form $M_1 \rightarrow M_2 + 2M_3$. The consumption rate of M_1 , due to this reaction, is v_1 ; the production rate of M_2 is v_1 ; and the production rate of M_3 is $2v_1$. (The reaction R_1 has no effect on metabolites M_4, \dots, M_m .) This corresponds to a first column of S of the form $(-1, 1, 2, 0, \dots, 0)$.

Reactions are also used to model flow of metabolites into and out of the cell. For example, suppose that reaction R_2 corresponds to the flow of metabolite M_1 into the cell, with v_2 giving the flow rate. (When $v_2 < 0$, it means that $|v_2|$ is the flow rate of the metabolite out of the cell.) This corresponds to a second column of S of the form $(1, 0, \dots, 0)$.

The last reaction, R_n , corresponds to biomass creation, or cell growth, so the reaction rate v_n is the cell growth rate. The last column of S gives the amounts of metabolites used (when the entry is negative) or created (when positive) per unit of cell growth rate.

Since our reactions include metabolites entering or leaving the cell, as well as those converted to biomass within the cell, we have conservation of the metabolites, which can be expressed as the *flux balance equation* $Sv = 0$.

Finally, we consider the effect of knocking out a gene. For simplicity, we'll assume that reactions $1, \dots, n - 1$ are each controlled by an associated gene, *i.e.*, gene G_k controls reaction R_k . Knocking out G_k has the effect of setting the associated reaction rate to zero.

Finally, we get to the point of all this. Suppose there is *no* $v \in \mathbf{R}^n$ that satisfies

$$Sv = 0, \quad v_k = 0, \quad v_n > 0.$$

This means there are no reaction rates consistent with cell growth, flux balance, and the gene knockout. In this case, we predict that knocking out gene G_k will kill the cell, and call gene G_k an *essential gene*.

- (a) Explain how to find all essential genes, given the stoichiometry matrix S . You can use any concepts from the class, *e.g.*, range, nullspace, least-squares.
- (b) Carry out your method for the problem data given in `fba_data.m`. List all essential genes.

Remark. This is a very simple version of the problem. In EE364a, you'll see more sophisticated versions of the same problem, that incorporate lower and upper limits on reactions rates and other realistic constraints.

5. *Portfolio selection with sector neutrality constraints.* We consider the problem of selecting a portfolio composed of n assets. We let $x_i \in \mathbf{R}$ denote the investment (say, in dollars) in asset i , with $x_i < 0$ meaning that we hold a short position in asset i . We normalize our total portfolio as $\mathbf{1}^T x = 1$, where $\mathbf{1}$ is the vector with all entries 1. (With normalization, the x_i are sometimes called *portfolio weights*.)

The portfolio (mean) return is given by $r = \mu^T x$, where $\mu \in \mathbf{R}^n$ is a vector of asset (mean) returns. We want to choose x so that r is large, while avoiding risk exposure, which we explain next.

First we explain the idea of *sector exposure*. We have a list of k economic sectors (such as manufacturing, energy, transportation, defense, ...). A matrix $F \in \mathbf{R}^{k \times n}$, called the *factor loading matrix*, relates the portfolio x to the *factor exposures*, given as $R^{\text{fact}} = Fx \in \mathbf{R}^k$. The number R_i^{fact} is the portfolio risk exposure to the i th economic sector. If R_i^{fact} is large (in magnitude) our portfolio is exposed to risk from changes in that sector; if it is small, we are less exposed to risk from that sector. If $R_i^{\text{fact}} = 0$, we say that the portfolio is *neutral* with respect to sector i .

Another type of risk exposure is due to fluctuations in the returns of the individual assets. The *idiosyncratic risk* is given by

$$R^{\text{id}} = \sum_{i=1}^n \sigma_i^2 x_i^2,$$

where $\sigma_i > 0$ are the standard deviations of the asset returns. (You can take the formula above as a definition; you do not need to understand the statistical interpretation.)

We will choose the portfolio weights x so as to maximize $r - \lambda R^{\text{id}}$, which is called the *risk-adjusted return*, subject to neutrality with respect to all sectors, *i.e.*, $R^{\text{fact}} = 0$. Of course we also have the normalization constraint $\mathbf{1}^T x = 1$. The parameter λ , which is positive, is called the *risk aversion parameter*. The (known) data in this problem are $\mu \in \mathbf{R}^n$, $F \in \mathbf{R}^{k \times n}$, $\sigma = (\sigma_1, \dots, \sigma_n) \in \mathbf{R}^n$, and $\lambda \in \mathbf{R}$.

- (a) Explain how to find x , using methods from the course. You are welcome (even encouraged) to express your solution in terms of block matrices, formed from the given data.
- (b) Using the data given in `sector_neutral_portfolio_data.m`, find the optimal portfolio. Report the associated values of r (the return), and R^{id} (the idiosyncratic risk). Verify that $\mathbf{1}^T x = 1$ (or very close) and $R^{\text{fact}} = 0$ (or very small).

6. *Memory of a linear time-invariant system.* An input signal (sequence) $u_t \in \mathbf{R}$, $t \in \mathbf{Z}$ (i.e., $t = \dots, -1, 0, 1, \dots$) and output signal $y_t \in \mathbf{R}$, $t \in \mathbf{Z}$, are related by a convolution operator

$$y_t = \sum_{\tau=1}^M h_{\tau} u_{t-\tau}, \quad t \in \mathbf{Z},$$

where $h = (h_1, \dots, h_M)$ are the *impulse response coefficients* of the convolution system. (Convolution systems are also called linear time-invariant systems.) When $h_M \neq 0$, the integer M is called the *memory* of the system.

Now for the problem. You are given the input and output signal values over a time interval $t = 1, \dots, T$:

$$(u_1, \dots, u_T), \quad (y_1, \dots, y_T).$$

The goal is to find the smallest memory M consistent with this data. Note that *you do not know u_{τ} or y_{τ} for $\tau \leq 0$ or $\tau > T$* , and of course, you do not know h .

- (a) Explain how to solve this problem, using any concepts from the course. You may assume that $T > 2M$.
- (b) Use your method from part (a) on the data found in `mem_lti_data.m`. Give the value of M found. *Hint.* You may find the matlab functions `toeplitz` and `fliplr` useful.