EE263 Prof. S. Boyd
Dec. 7–8 or Dec. 8–9, 2007.

Final exam

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

Please read the following instructions carefully.

- You may use any books, notes, or computer programs (*e.g.*, Matlab), but you may not discuss the exam with anyone until Dec. 10, 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. We've tried pretty hard to make the exam unambiguous and clear, so we're unlikely to say much.

- Please address email inquiries to `ee263-aut0708-staff@lists.stanford.edu`. This forwards the mail to the professor and the TAs. In particular, please do not use Stephen Boyd's or the TAs' individual email addresses.

- 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) to your exam, and assemble your solutions to the problems in order, *i.e.*, problem 1, problem 2, ..., problem 7. Start each solution on a new page. Do not collect all plots (for example) at the end of the exam; plots for problem 3 (say) should be with your solution to problem 3.

- 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 with 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 (what appears to be) a practical setting. *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/final_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.*

- Problems may be easier (or harder) than they first appear. None of the problems requires long calculations or any serious programming. Some may involve slight extensions of, or variations on, material we have covered.

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

- **IMPORTANT**: Here are a few Matlab hints specific to this exam: The commands `orth` and `null` may be useful to you. (Or, maybe not.) To compute the standard (*i.e.*, not full) SVD of a matrix A, use `svd(A,'econ')`.

1. *Optimal initial conditions for a bioreactor.* The dynamics of a bioreactor are given by $\dot{x}(t) = Ax(t)$, where $x(t) \in \mathbf{R}^n$ is the state, with $x_i(t)$ representing the total mass of species or component i at time t. Component i has (positive) value (or cost) c_i, so the total value (or cost) of the components at time t is $c^T x(t)$. (We ignore any extra cost that would be incurred in separating the components.) Your job is to choose the initial state, under a budget constraint, that maximizes the total value at time T. More specifically, you are to choose $x(0)$, with all entries nonnegative, that satisfies $c^T x(0) \leq B$, where B is a given positive budget. The problem data (*i.e.*, things you know) are A, c, T, and B.

   You can assume that A is such that, for any $x(0)$ with nonnegative components, $x(t)$ will also have all components nonnegative, for any $t \geq 0$. (This occurs, by the way, if and only if the off-diagonal entries of A are nonnegative.)

   (a) Explain how to solve this problem.

   (b) Carry out your method on the specific instance with data

$$A = \begin{bmatrix} 0.1 & 0.1 & 0.3 & 0 \\ 0 & 0.2 & 0.4 & 0.3 \\ 0.1 & 0.3 & 0.1 & 0 \\ 0 & 0 & 0.2 & 0.1 \end{bmatrix}, \quad c = \begin{bmatrix} 3.5 \\ 0.6 \\ 1.1 \\ 2.0 \end{bmatrix}, \quad T = 10, \quad B = 1.$$

   Give the optimal $x(0)$, and the associated (optimal) terminal value $c^T x(T)$.

   Give us the terminal value obtained when the initial state has equal mass in each component, *i.e.*, $x(0) = \alpha \mathbf{1}$, with α adjusted so that the total initial cost is B. Compare this with the optimal terminal value.

   Also give us the terminal value obtained when the same amount, B/n, is spent on each initial state component (*i.e.*, $x(0)_i = B/(nc_i)$). Compare this with the optimal terminal value.

2. *Simultaneously estimating student ability and exercise difficulty.* Each of n students takes an exam that contains m questions. Student j receives (nonnegative) grade G_{ij} on question i. One simple model for predicting the grades is to estimate $G_{ij} \approx \hat{G}_{ij} = a_j/d_i$, where a_j is a (nonnegative) number that gives the *ability* of student j, and d_i is a (positive) number that gives the *difficulty* of exam question i. Given a particular model, we could simultaneously scale the student abilities and the exam difficulties by any positive number, without affecting $\hat{G}_{ij}$. Thus, to ensure a unique model, we will *normalize* the exam question difficulties d_i, so that the mean exam question difficulty across the m questions is 1.

   In this problem, you are given a complete set of grades (*i.e.*, the matrix $G \in \mathbf{R}^{m\times n}$). Your task is to find a set of nonnegative student abilities, and a set of positive, normalized question difficulties, so that $G_{ij} \approx \hat{G}_{ij}$. In particular, choose your model to minimize the RMS error, J,

$$J = \left(\frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(G_{ij} - \hat{G}_{ij} \right)^2 \right)^{1/2}.$$

   This can be compared to the RMS value of the grades,

$$\left(\frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} G_{ij}^2 \right)^{1/2}.$$

   (a) Explain how to solve this problem, using any concepts from EE263. If your method is approximate, or not guaranteed to find the global minimum value of J, say so. If carrying out your method requires some rank or other conditions to hold, say so.

   Note: You do not have to concern yourself with the requirement that a_j are nonnegative and d_i are positive. You can just assume this works out, or is easily corrected.

   (b) Carry out your method on the data found in `grademodeldata.m`. Give the optimal value of J, and also express it as a fraction of the RMS value of the grades. Give the difficulties of the 7 problems on the exam.

3. *Optimal espresso cup pre-heating.* At time $t = 0$ boiling water, at 100°C, is poured into an espresso cup; after P seconds (the 'pre-heating time'), the water is poured out, and espresso, with initial temperature 95°C, is poured in. (You can assume this operation occurs instantaneously.) The espresso is then consumed exactly 15 seconds later (yes, instantaneously). The problem is to choose the pre-heating time P so as to maximize the temperature of the espresso when it is consumed.

   We now give the thermal model used. We take the temperature of the liquid in the cup (water or espresso) as one state; for the cup we use an n-state finite element model. The vector $x(t) \in \mathbf{R}^{n+1}$ gives the temperature distribution at time t: $x_1(t)$ is the liquid (water or espresso) temperature at time t, and $x_2(t), \ldots, x_{n+1}(t)$ are the temperatures of the elements in the cup. All of these are in degrees C, with t in seconds. The dynamics are

$$\frac{d}{dt}(x(t) - 20 \cdot \mathbf{1}) = A(x(t) - 20 \cdot \mathbf{1}),$$

   where $A \in \mathbf{R}^{(n+1)\times(n+1)}$. (The vector $20 \cdot \mathbf{1}$, with all components 20, represents the ambient temperature.) The initial temperature distribution is

$$x(0) = \begin{bmatrix} 100 \\ 20 \\ \vdots \\ 20 \end{bmatrix}.$$

   At $t = P$, the liquid temperature changes instantly from whatever value it has, to 95; the other states do not change. Note that the dynamics of the system are the same before and after pre-heating (because we assume that water and espresso behave in the same way, thermally speaking).

   We have *very generously* derived the matrix A for you. You will find it in `espressodata.m`. In addition to `A`, the file also defines `n`, and, respectively, the ambient, espresso and preheat water temperatures `Ta` (which is 20), `Te` (95), and `Tl` (100).

   Explain your method, submit your code, and give final answers, which must include the optimal value of P and the resulting optimal espresso temperature when it is consumed. Give both to an accuracy of one decimal place, as in

   '$P = 23.5$ s, which gives an espresso temperature at consumption of 62.3°C.'

   (This is not the correct answer, of course.)

4. *Optimal dynamic purchasing.* You are to complete a large order to buy a certain number, B, of shares in some company. You are to do this over T time periods. (Depending on the circumstances, a single time period could be between tens of milliseconds and minutes.) We will let b_t denote the number of shares bought in time period t, for $t = 1, \ldots, T$, so we have $b_1 + \cdots + b_T = B$. (The quantities $B, b_1, \ldots, b_T$ can all be any real number; $b_t < 0$, for example, means we *sold* shares in the period t. We also don't require b_t to be integers.) We let p_t denote the price per share in period t, so the total cost of purchasing the B shares is $C = p_1 b_1 + \cdots + p_T b_T$.

   The amounts we purchase are large enough to have a noticeable effect on the price of the shares. The prices change according to the following equations:

$$p_1 = \bar{p} + \alpha b_1, \qquad p_t = \theta p_{t-1} + (1-\theta)\bar{p} + \alpha b_t, \quad t = 2, \ldots, T.$$

   Here $\bar{p}$ is the base price of the shares and α and θ are parameters that determine how purchases affect the prices. The parameter α, which is positive, tells us how much the price goes up in the current period when we buy one share. The parameter θ, which lies between 0 and 1, measures the *memory*: If $\theta = 0$ the share price has no memory, and the purchase made in period t only affects the price in that period; if θ is 0.5 (say), the effect a purchase has on the price decays by a factor of two between periods. If $\theta = 1$, the price has perfect memory and the price change will persist for all future periods.

   If purchases didn't increase the price, the cost of purchasing the shares would always be $\bar{p}B$. The difference between the total cost and this cost, $C - \bar{p}B$, is called the *transaction cost*.

   Find the purchase quantities $b_1, \ldots, b_T$ that minimize the transaction cost $C - \bar{p}B$, for the particular problem instance with

$$B = 10000, \quad T = 10, \quad \bar{p} = 10, \quad \theta = 0.8, \quad \alpha = 0.00015.$$

   Give the optimal transaction cost. Also give the transaction cost if all the shares were purchased in the first period, and the transaction cost if the purchases were evenly spread over the periods (*i.e.*, if 1000 shares were purchased in each period). Compare these three quantities.

   You must explain your method clearly, using any concepts from this class, such as least-squares, pseudo-inverses, eigenvalues, singular values, etc. If your method requires that some rank or other conditions to hold, say so. You must also check, in your Matlab code, that these conditions are satisfied for the given problem instance.

5. *Angle between two subspaces.* The angle between two nonzero vectors v and w in $\mathbf{R}^n$ is defined as

$$\angle(v, w) = \cos^{-1}\left(\frac{v^T w}{\|v\|\|w\|}\right),$$

where we take $\cos^{-1}(a)$ as being between 0 and π. We define the angle between a nonzero vector $v \in \mathbf{R}^n$ and a (nonzero) subspace $\mathcal{W} \subseteq \mathbf{R}^n$ as

$$\angle(v, \mathcal{W}) = \min_{w \in \mathcal{W},\, w \neq 0} \angle(v, w).$$

Thus, $\angle(v, \mathcal{W}) = 10^\circ$ means that the smallest angle between v and any vector in $\mathcal{W}$ is 10°. If $v \in \mathcal{W}$, we have $\angle(v, \mathcal{W}) = 0$.

Finally, we define the angle between two nonzero subspaces $\mathcal{V}$ and $\mathcal{W}$ as

$$\angle(\mathcal{V}, \mathcal{W}) = \max\left\{\max_{v \in \mathcal{V},\, v \neq 0} \angle(v, \mathcal{W}), \max_{w \in \mathcal{W},\, w \neq 0} \angle(w, \mathcal{V})\right\}.$$

This angle is zero if and only if the two subspaces are equal. If $\angle(\mathcal{V}, \mathcal{W}) = 10^\circ$, say, it means that either there is a vector in $\mathcal{V}$ whose minimum angle to any vector of $\mathcal{W}$ is 10°, or there is a vector in $\mathcal{W}$ whose minimum angle to any vector of $\mathcal{V}$ is 10°.

(a) Suppose you are given two matrices $A \in \mathbf{R}^{n \times r}$, $B \in \mathbf{R}^{n \times r}$, each of rank r. Let $\mathcal{V} = \text{range}(A)$ and $\mathcal{W} = \text{range}(B)$. Explain how you could find or compute $\angle(\mathcal{V}, \mathcal{W})$. You can use any of the concepts in the class, *e.g.*, least-squares, QR factorization, pseudo-inverse, norm, SVD, Jordan form, etc.

(b) Carry out your method for the matrices found in `angsubdata.m`. Give the numerical value for $\angle(\text{range}(A), \text{range}(B))$.

6. *Extracting the faintest signal.* An n-vector valued signal, $x(t) \in \mathbf{R}^n$, is defined for $t = 1, \ldots, T$. We'll refer to its ith component, $x_i(t)$, for $t = 1, \ldots, T$, as the ith scalar signal. The scalar signals $x_1, \ldots, x_{n-1}$ have an RMS value substantially larger than x_n. In other words, x_n is the faintest scalar signal. It is also the signal of interest for this problem. We will assume that the scalar signals $x_1, \ldots, x_n$ are unrelated to each other, and so are nearly uncorrelated (*i.e.*, nearly orthogonal).

   We aren't given the vector signal $x(t)$, but we are given a linear transformation of it,

   $$y(t) = Ax(t), \quad t = 1, \ldots, T,$$

   where $A \in \mathbf{R}^{n \times n}$ is invertible. If we knew A, we could easily recover the original signal (and therefore also the faintest scalar signal $x_n(t)$), using $x(t) = A^{-1}y(t)$, $t = 1, \ldots, T$. But, sadly, we don't know A.

   Here is a heuristic method for guessing $x_n(t)$. We will form our estimate as

   $$\hat{x}_n(t) = w^T y(t), \quad t = 1, \ldots, T,$$

   where $w \in \mathbf{R}^n$ is a vector of weights. Note that if w were chosen so that $w^T A = \alpha e_n^T$, with $\alpha \neq 0$ a constant, then we would have $\hat{x}_n(t) = \alpha x_n(t)$, *i.e.*, a perfect reconstruction except for the scale factor α.

   Now, the important part of our heuristic: we choose w to minimize the RMS value of $\hat{x}_n$, subject to $\|w\| = 1$. *Very roughly*, one idea behind the heuristic is that, in general, $w^T y$ is a linear combination of the scalar signals $x_1, \ldots, x_n$. If the linear combination has a small norm, that's because the linear combination is 'rich in x_n', and has only a small amount of energy contributed by $x_1, \ldots, x_{n-1}$. That, in fact, is exactly what we want. In any case, you don't need to worry about why the heuristic works (or doesn't work)—it's the method you are going to use in this problem.

   (a) Explain how to find a w that minimizes the RMS value of $\hat{x}_n$, using concepts from the class (*e.g.*, range, rank, least-squares, QR factorization, eigenvalues, singular values, and so on).

   (b) Carry out your method on the problem instance with $n = 4$, $T = 26000$, described in the Matlab file `faintestdata.m`. This file will define an $n \times T$ matrix Y, where the tth column of Y is the vector $y(t)$. The file will also define n and T. Submit your code, and give us the optimal weight vector $w \in \mathbf{R}^4$ you find, along with the associated RMS value of $\hat{x}_n$.

   The following is not needed to solve the problem. The signals are actually audio tracks, each 3.25 seconds long and sampled at 8 kHz. The Matlab file `faintestaudio.m` contains commands to generate wave files of the linear combinations $y_1, \ldots, y_4$, and a wave file of your estimate $\hat{x}_n$. You are welcome to generate and listen to these files.

7. *Some true-false questions.* In the following statements, $A \in \mathbf{R}^{n\times n}$, $\sigma_{\min}$ refers to σ_n (the nth largest singular value), and κ refers to the condition number. Tell us whether each statement is true or false. 'True' means that the statement holds for any matrix $A \in \mathbf{R}^{n\times n}$, for any n. 'False' means that the statement is not true. The only answers we will read are 'True', 'False', and 'My attorney has advised me to not answer this question at this time'. (This last choice will receive partial credit.) If you write anything else, you will receive no credit for that statement. In particular, do not write justification for any answer, or provide any counter-examples.

   (a) $\|e^A\| \leq e^{\|A\|}$.

   (b) $\sigma_{\min}(e^A) \geq e^{\sigma_{\min}(A)}$.

   (c) $\kappa(e^A) \leq e^{\kappa(A)}$.

   (d) $\kappa(e^A) \leq e^{2\|A\|}$.

   (e) $\mathbf{Rank}(e^A) \geq \mathbf{Rank}(A)$.

   (f) $\mathbf{Rank}(e^A - I) \leq \mathbf{Rank}(A)$.