EE263 Prof. S. Boyd
Dec. 8–9 or Dec. 9–10, 2006.

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. 12, 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-aut0607-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 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 networking, equalizer design, graph theory, and control. *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.

- Please be careful to avoid committing crimes against matrices, as outlined in the notes on the course web page. Special penalties will apply for any of these crimes committed during the final exam.

- Four 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/matlab/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.

- 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.

1. *Analysis and optimization of a communication network.* A communication network is modeled as a set of m directed links connecting nodes. There are n *routes* in the network. A route is a path, along one or more links in the network, from a *source node* to a *destination node*. In this problem, the routes are fixed, and are described by an $m \times n$ *route-link matrix* A, defined as

$$A_{ij} = \begin{cases} 1 & \text{route } j \text{ passes through link } i \\ 0 & \text{otherwise.} \end{cases}$$

Over each route we have a nonnegative *flow*, measured in (say) bits per second. We denote the flow along route j as f_j, and we call $f \in \mathbf{R}^n$ the *flow vector*. The *traffic* on a link i, denoted t_i, is the sum of the flows on all routes passing through link i. The vector $t \in \mathbf{R}^m$ is called the *traffic vector*.

Each link has an associated nonnegative *delay*, measured in (say) seconds. We denote the delay for link i as d_i, and refer to $d \in \mathbf{R}^m$ as the *link delay vector*. The *latency* on a route j, denoted l_j, is the sum of the delays along each link constituting the route, *i.e.*, the time it takes for bits entering the source to emerge at the destination. The vector $l \in \mathbf{R}^n$ is the *route latency vector*.

The total number of bits in the network at an instant in time is given by $B = f^T l = t^T d$.

(a) *Worst-case flows and delays.* Suppose the flows and link delays satisfy

$$(1/n)\sum_{j=1}^{n} f_j^2 \leq F^2, \qquad (1/m)\sum_{i=1}^{m} d_i^2 \leq D^2,$$

where F and D are given. What is the maximum possible number of bits in the network? What values of f and d achieve this maximum value? (For this problem you can ignore the constraint that the flows and delays must be nonnegative. It turns out, however, that the worst-case flows and delays can always be chosen to be nonnegative.)

(b) *Utility maximization.* For a flow f_j, the network operator derives income at a rate $p_j f_j$, where p_j is the price per unit flow on route j. The network operator's total rate of income is thus $\sum_{j=1}^{n} p_j f_j$. (The route prices are known and positive.)

The network operator is charged at a rate $c_i t_i$ for having traffic t_i on link i, where c_i is the cost per unit of traffic on link i. The total charge rate for link traffic is $\sum_{i=1}^{m} t_i c_i$. (The link costs are known and positive.) The net income rate (or utility) to the network operator is therefore

$$U^{\text{net}} = \sum_{j=1}^{n} p_j f_j - \sum_{i=1}^{m} c_i t_i.$$

Find the flow vector f that maximizes the operator's net income rate, subject to the constraint that each f_j is between 0 and $F^{\max}$, where $F^{\max}$ is a given positive maximum flow value.

2. *Stability of a time-varying system.* We consider a discrete-time linear dynamical system

$$x(t+1) = A(t)x(t),$$

where $A(t) \in \{A_1, A_2, A_3, A_4\}$. These 4 matrices, which are 4×4, are given in `tv_data.m`.

Show that this system is stable, *i.e.*, for any trajectory x, we have $x(t) \to 0$ as $t \to \infty$. (This means that for any $x(0)$, and for any sequence $A(0), A(1), A(2), \ldots$, we have $x(t) \to 0$ as $t \to \infty$.)

You may use any methods or concepts used in the class, *e.g.*, least-squares, eigenvalues, singular values, controllability, and so on. Your proof will consist of two parts:

- An explanation of how you are going to show that any trajectory converges to zero. Your argument of course will require certain conditions (that you will find) to hold for the given data $A_1, \ldots, A_4$.
- The numerical calculations that verify the conditions hold for the given data. You must provide the source code for these calculations, and show the results as well.

Your answer is limited to three pages, including the numerical verification. We will not read past three pages. Of course, you are welcome to submit a solution that is *shorter* than three pages!

3. *Some bounds on singular values.* Suppose $A \in \mathbf{R}^{6\times 3}$, with singular values 7, 5, 3, and $B \in \mathbf{R}^{6\times 3}$, with singular values 2, 2, 1. Let $C = [A\ B] \in \mathbf{R}^{6\times 6}$, with full SVD $C = U\Sigma V^T$, with $\Sigma = \mathbf{diag}(\sigma_1, \ldots, \sigma_6)$. (We allow the possibility that some of these singular values are zero.)

   (a) How large can σ_1 be?

   (b) How small can σ_1 be?

   (c) How large can σ_6 be?

   (d) How small can σ_6 be?

   What we mean is, how large (or small) can the specified quantity be, for any A and B with the given sizes and given singular values.

   Please give only the answers, as specific numbers, with 3 digits after the decimal place. We're looking for answers that have the form

   $$(a)\ 12.420, \qquad (b)\ 10.000, \qquad (c)\ 0.552, \qquad (d)\ 0.000$$

   (This is just an example). We will not read *any* derivation or justification. You do not have to find A and B that achieve the values you give.

4. *Optimal choice of initial temperature profile.* We consider a thermal system described by an n-element finite-element model. The elements are arranged in a line, with the temperature of element i at time t denoted $T_i(t)$. Temperature is measured in degrees Celsius above ambient; negative $T_i(t)$ corresponds to a temperature below ambient. The dynamics of the system are described by

$$c_1\dot{T}_1 = -a_1T_1 - b_1(T_1 - T_2),$$

$$c_i\dot{T}_i = -a_iT_i - b_i(T_i - T_{i+1}) - b_{i-1}(T_i - T_{i-1}), \quad i = 2, \ldots, n-1,$$

and

$$c_n\dot{T}_n = -a_nT_n - b_{n-1}(T_n - T_{n-1}).$$

where $c \in \mathbf{R}^n$, $a \in \mathbf{R}^n$, and $b \in \mathbf{R}^{n-1}$ are given and are all positive.

We can interpret this model as follows. The parameter c_i is the heat capacity of element i, so $c_i\dot{T}_i$ is the net heat flow into element i. The parameter a_i gives the thermal conductance between element i and the environment, so a_iT_i is the heat flow from element i to the environment (*i.e.*, the direct heat loss from element i.) The parameter b_i gives the thermal conductance between element i and element $i+1$, so $b_i(T_i - T_{i+1})$ is the heat flow from element i to element $i+1$. Finally, $b_{i-1}(T_i - T_{i-1})$ is the heat flow from element i to element $i-1$.

The goal of this problem is to choose the initial temperature profile, $T(0) \in \mathbf{R}^n$, so that $T(t^{\text{des}}) \approx T^{\text{des}}$. Here, $t^{\text{des}} \in \mathbf{R}$ is a specific time when we want the temperature profile to closely match $T^{\text{des}} \in \mathbf{R}^n$. We also wish to satisfy a constraint that $\|T(0)\|$ should be not be too large.

To formalize these requirements, we use the objective $(1/\sqrt{n})\|T(t^{\text{des}}) - T^{\text{des}}\|$ and the constraint $(1/\sqrt{n})\|T(0)\| \leq T^{\text{max}}$. The first expression is the RMS temperature deviation, at $t = t^{\text{des}}$, from the desired value, and the second is the RMS temperature deviation from ambient at $t = 0$. T^{max} is the (given) maximum inital RMS temperature value.

(a) Explain how to find $T(0)$ that minimizes the objective while satisfying the constraint.

(b) Solve the problem instance with the values of n, c, a, b, t_{des}, T^{des} and T^{max} defined in the file `temp_prof_data.m`.

Plot, on one graph, your $T(0)$, $T(t^{\text{des}})$ and T^{des}. Give the RMS temperature error $(1/\sqrt{n})\|T(t^{\text{des}}) - T^{\text{des}}\|$, and the RMS value of initial temperature $(1/\sqrt{n})\|T(0)\|$.

5. *A heuristic for MAXCUT.* Consider a graph with n nodes and m edges, with the nodes labeled $1, \ldots, n$ and the edges labeled $1, \ldots, m$. We partition the nodes into two groups, B and C, *i.e.*, $B \cap C = \emptyset$, $B \cup C = \{1, \ldots, n\}$. We define the number of *cuts* associated with this partition as the number of edges between pairs of nodes when one of the nodes is in B and the other is in C. A famous problem, called the MAXCUT problem, involves choosing a partition (*i.e.*, B and C) that maximizes the number of cuts for a given graph. For any partition, the number of cuts can be no more than m. If the number of cuts is m, nodes in group B connect only to nodes in group C and the graph is bipartite.

   The MAXCUT problem has many applications. We describe one here, although you do not need it to solve this problem. Suppose we have a communication system that operates with a two-phase clock. During periods $t = 0, 2, 4, \ldots$, each node in group B transmits data to nodes in group C that it is connected to; during periods $t = 1, 3, 5, \ldots$, each node in group C transmits to the nodes in group B that it is connected to. The number of cuts, then, is exactly the number of successful transmissions that can occur in a two-period cycle. The MAXCUT problem is to assign nodes to the two groups so as to maximize the overall efficiency of communication.

   It turns out that the MAXCUT problem is hard to solve exactly, at least if we don't want to resort to an exhaustive search over all, or most of, the 2^{n-1} possible partitions. In this problem we explore a sophisticated heuristic method for finding a good (if not the best) partition in a way that scales to large graphs.

   We will encode the partition as a vector $x \in \mathbf{R}^n$, with $x_i \in \{-1, 1\}$. The associated partition has $x_i = 1$ for $i \in B$ and $x_i = -1$ for $i \in C$. We describe the graph by its node adjacency matrix $A \in \mathbf{R}^{n \times n}$ with

$$A_{ij} = \begin{cases} 1 & \text{there is an edge between node } i \text{ and node } j \\ 0 & \text{otherwise} \end{cases}$$

   Note that A is symmetric and $A_{ii} = 0$ (since we do not have self-loops in our graph).

   (a) Find a symmetric matrix P, with $P_{ii} = 0$ for $i = 1, \ldots, n$, and a constant d, for which $x^T P x + d$ is the number of cuts encoded by any partitioning vector x. Explain how to calculate P and d from A. Of course, P and d cannot depend on x.

   The MAXCUT problem can now be stated as the optimization problem

$$\begin{array}{ll} \text{maximize} & x^T P x + d \\ \text{subject to} & x_i^2 = 1, \quad i = 1, \ldots, n, \end{array}$$

   with variable $x \in \mathbf{R}^n$.

   (b) A famous heuristic for approximately solving MAXCUT is to replace the n constraints $x_i^2 = 1$, $i = 1, \ldots, n$, with a single constraint $\sum_{i=1}^n x_i^2 = n$, creating the

so-called *relaxed* problem

$$\begin{array}{ll} \text{maximize} & x^T P x + d \\ \text{subject to} & \sum_{i=1}^n x_i^2 = n. \end{array}$$

Explain how to solve this relaxed problem (even if you could not solve part (a)).

Let $x^\star$ be a solution to the relaxed problem. We generate our candidate partition with $x_i = \mathbf{sign}(x_i^\star)$. (This means that $x_i = 1$ if $x_i^\star \geq 0$, and $x_i = -1$ if $x_i^\star < 0$.)

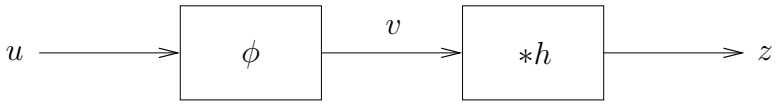
Remark: We can give a geometric interpretation of the relaxed problem, which will also explain why it's called relaxed. The constraints in the problem in part (a), that $x_i^2 = 1$, require x to lie on the vertices of the unit hypercube. In the relaxed problem, the constraint set is the unit ball of unit radius. Because this constraint set is larger than the original constraint set (*i.e.*, it includes it), we say the constraints have been relaxed.

(c) Run the MAXCUT heuristic described in part (b) on the data given in `mc_data.m`. How many cuts does your partition yield?

A simple alternative to MAXCUT is to generate a large number of random partitions, using the random partition that maximizes the number of cuts as an approximate solution. Carry out this method with 1000 random partitions generated by `x = sign(rand(n,1)-0.5)`. What is the largest number of cuts obtained by these random partitions?

Note: There are many other heuristics for approximately solving the MAXCUT problem. However, we are not interested in them. In particular, please do not submit any other method for approximately solving MAXCUT.

6. *Designing a nonlinear equalizer from I/O data.* This problem concerns the discrete-time system shown below, which consists of a memoryless nonlinearity ϕ, followed by a convolution filter with finite impulse response h. The scalar signal u is the input, and the scalar signal z is the output.



What this means is

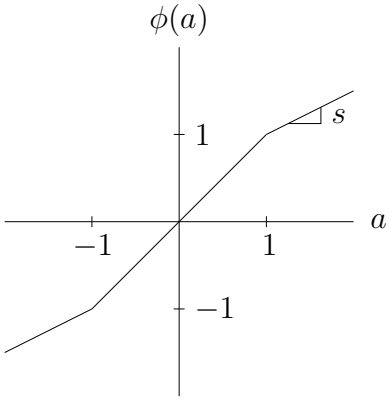
$$z(t) = \sum_{\tau=0}^{M-1} h(\tau)v(t-\tau), \qquad v(t) = \phi(u(t)), \qquad t \in \mathbf{Z}.$$

(Note that these signals are defined for all integer times, not just nonnegative times.)

Here $\phi : \mathbf{R} \to \mathbf{R}$, with the specific form

$$\phi(a) = \begin{cases} a & -1 \leq a \leq 1 \\ 1 - s + sa & a > 1 \\ -1 + s + sa & a < -1, \end{cases}$$

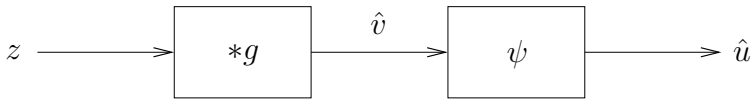
where $s > 0$ is a parameter. This function is shown below.



Here is an interpretation (that is not needed to solve the problem). The nonlinear function ϕ represents a power amplifier that is nonlinear for input signals larger than one in magnitude; s is called the *saturation gain* of the amplifier. The convolution system represents the transmission channel.

We are going to design an *equalizer* for the system, *i.e.*, another system that takes the signal z as input, and gives an output $\hat{u}$ which is an approximation of the input signal u.

Our equalizer will have the form shown below.



This means

$$\hat{v}(t) = \sum_{\tau=0}^{M-1} g(\tau) z(t-\tau), \qquad \hat{u}(t) = \psi(\hat{v}(t)), \qquad t \in \mathbf{Z}.$$

This equalizer will work well provided $g * h \approx \delta$ (in which case $\hat{v}(t) \approx v(t)$), and $\psi = \phi^{-1}$ (*i.e.*, $\psi(\phi(a)) = a$ for all a).

To make sure our (standard) notation here is clear, recall that

$$(g * h)(t) = \sum_{\tau=\max\{0,t-M+1\}}^{\min\{M-1,t\}} g(\tau) h(t-\tau), \quad t = 0, \ldots, 2M-1.$$

(Note: in matlab `conv(g,h)` gives the convolution of `g` and `h`, but these vectors are indexed from 1 to M, *i.e.*, `g(1)` corresponds to $g(0)$.) The term δ is the Kronecker delta, defined as $\delta(0) = 1$, $\delta(i) = 0$ for $i \neq 0$.

Now, finally, we come to the problem. You are given some input/output (I/O) data $u(1), \ldots, u(N)$, $z(1), \ldots, z(N)$, and M (the length of g, and also the length of h). You do *not* know the parameter s, or the channel impulse response $h(0), \ldots, h(M-1)$. You also don't know $u(t)$, $z(t)$ for $t \leq 0$.

(a) Explain how to find $\hat{s}$, an estimate of the saturation gain s, and $g(0), \ldots, g(M-1)$, that minimize

$$J = \frac{1}{N-M+1} \sum_{i=M}^{N} \Big(\hat{v}(i) - \phi(u(i))\Big)^2.$$

Here u refers to the given input data, and $\hat{v}$ comes from the given output data z. Note that if $g * h = \delta$ and $s = \hat{s}$, we have $J = 0$.

We exclude $i = 1, \ldots, M-1$ in the sum defining J because these terms depend (through $\hat{v}$) on $z(0), z(-1), \ldots$, which are unknown.

(b) Apply your method to the data given in the file `nleq_data.m`. Give the values of the parameters $\hat{s}$ and $g(0), \ldots, g(M-1)$ found, as well as J. Plot g using the matlab command `stem`.

(c) Using the values of $\hat{s}$ and $g(0), \ldots, g(M-1)$ found in part (b), find the equalized signal $\hat{u}(t)$, for $t = 1, \ldots, N$. For the purposes of finding $\hat{u}(t)$ you can assume that $z(t) = 0$ for $t \leq 0$. As a result, we can expect a large equalization error (*i.e.*, $\hat{u}(t) - u(t)$) for $t = 1, \ldots, M-1$.

Plot the input signal $u(t)$, the output signal $z(t)$, the equalized signal $\hat{u}(t)$, and the equalization error $\hat{u}(t) - u(t)$, for $t = 1, \ldots, N$.

7. *A greedy control scheme.* Our goal is to choose an input $u : \mathbf{R}_+ \to \mathbf{R}^m$, that is not too big, and drives the state $x : \mathbf{R}_+ \to \mathbf{R}^n$ of the system $\dot{x} = Ax + Bu$ to zero quickly. To do this, we will choose $u(t)$, for each t, to minimize the quantity

$$\frac{d}{dt}\|x(t)\|^2 + \rho\|u(t)\|^2,$$

where $\rho > 0$ is a given parameter. The first term gives the rate of decrease (if it is negative) of the norm-squared of the state vector; the second term is a penalty for using a large input.

This scheme is greedy because at each instant t, $u(t)$ is chosen to minimize the compositive objective above, without regard for the effects such an input might have in the future.

(a) Show that $u(t)$ can be expressed as $u(t) = Kx(t)$, where $K \in \mathbf{R}^{m\times n}$. Give an explicit formula for K. (In other words, the control scheme has the form of a constant linear state feedback.)

(b) What are the conditions on A, B, and ρ under which we have $(d/dt)\|x(t)\|^2 < 0$ whenever $x(t) \neq 0$, using the scheme described above? (In other words, when does this control scheme result in the norm squared of the state always decreasing?)

(c) Find an example of a system (*i.e.*, A and B), for which the open-loop system $\dot{x} = Ax$ is stable, but the closed-loop system $\dot{x} = Ax + Bu$ (with u as above) is unstable, when $\rho = 1$. Try to find the simplest example you can, and be sure to show us verification that the open-loop system is stable and that the closed-loop system is not. (We will *not* check this for you. You must explain how to check this, and attach code and associated output.)