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

Final exam solutions

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.

Solution: The traffic vector t is given by $t = Af$, and the route latency vector l is given by $l = A^T d$.

(a) The number of bits in the network is given by $B = d^T Af$. The problem, then, is to find

$$B^{\max} = \max_{\|f\|^2 \leq nF^2,\ \|d\|^2 \leq mD^2} d^T Af.$$

We know that, for any f and d satisfying the constraints, we have

$$d^T Af \leq \|d\|\|A\|\|f\| \leq FD\sqrt{mn}\|A\|.$$

It follows that $B^{\max} \leq FD\sqrt{mn}\|A\|$. We are now going to show that equality holds here.

Consider the choices

$$f = F\sqrt{n}v_{\max}, \qquad d = D\sqrt{m}u_{\max},$$

where $v_{\max}$ and $u_{\max}$ are, respectively, the right and left singular vectors with unit norm, associated with $\|A\|$, the largest singular value of A. For these choices we have

$$d^T Af = FD\sqrt{mn}u_{\max}^T Av_{\max} = FD\sqrt{mn}u_{\max}^T\|A\|u_{\max} = FD\sqrt{mn}\|A\|.$$

The maximum number of bits in the network is therefore

$$B^{\max} = FD\sqrt{mn}\|A\|.$$

(b) Let $p \in \mathbf{R}^n$ be the vector whose jth entry is p_j and let $c \in \mathbf{R}^m$ be the vector whose ith entry is c_i. The net income rate of the network is given by

$$U^{\text{net}} = p^T f - c^T t = p^T f - c^T Af = (p - A^T c)^T f.$$

We need to find the flow distribution f that solves the following problem:

$$\begin{array}{ll} \text{maximize} & (p - A^T c)^T f \\ \text{subject to} & 0 \leq f \leq F^{\max}. \end{array}$$

Now it *looks* like the Cauchy-Schwarz inequality should be useful here, since we are choosing a vector (f) to maximize its inner product with another vector ($p - A^T c$). However, the constraint on f is *not* on its norm, so Cauchy-Schwarz isn't relevant here.

Defining $w = p - A^T c$, the objective is to maximize $\sum_i w_i f_i$ subject to $0 \leq f_i \leq F^{\max}$. We can choose the f_i's separately (since the constraints involve each f_i separately, and the objective is a sum of terms, each involving only f_i). If w_i is negative, it means that the costs for route i exceed income, so the best we can do is set $f_i = 0$. If $w_i > 0$, which means that the revenue exceeds the cost, we should set $f_i = F^{\max}$.

In summary, the solution is given by

$$f_j = \begin{cases} F^{\max} & w_j > 0 \\ 0 & w_j \leq 0 \end{cases}$$

for $j = 1, \ldots, p$.

We can give a very simple interpretation of this optimal flow. The number w_j is exactly the revenue per unit flow j (*i.e.*, p_j) minus the total link cost per unit flow over all the links that route j goes over (*i.e.*, $(A^T c)_j$). If this net number is positive, then we set the flow to its maximum value (in order to get the maximum net utility). If this net number is negative, we set the flow to zero.

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!

Solution.
First some discussion. We can check that each of the matrices A_i is stable, *i.e.*, all eigenvalues have magnitude less than one. Of course this has to be the case, because a possible sequence is $A(t) = A_i$ for all t. But this certainly does not prove that the time-varying system is stable. Simple examples show that time-varying systems can be unstable, even when for each t, $A(t)$ is stable.

As far as we know, *no* argument based on the magnitude of eigenvalues (of *any* matrix or matrices) is correct. Basically, eigenvalues were just not relevant in this problem.

Instead, we are going to argue that the norm of the state decreases to zero. We find that the maximum gains (matrix norms) of the matrices are:

```
norm(A1) = 6.016509
norm(A2) = 5.007023
norm(A3) = 1.999188
norm(A4) = 0.149993
```

This shows that multiplication by A_4 does indeed reduce the norm of a vector, by a factor less than one. But the matrices A_1, A_2, and A_3 can amplify the norm of a vector by factors much larger than one. In particular, it's not true that the norm of $x(t)$ decreases at each step. But we note, for future use, that $\|x(t+1)\| \leq 6.02\|x(t)\|$.

Now we examine the gain of products of *pairs* of the matrices.

```
norm(A1*A1) = 0.163270
norm(A1*A2) = 0.498489
norm(A1*A3) = 0.907308
norm(A1*A4) = 0.836615
norm(A2*A1) = 0.149617
norm(A2*A2) = 0.253311
norm(A2*A3) = 0.754820
norm(A2*A4) = 0.696261
norm(A3*A1) = 0.484295
norm(A3*A2) = 0.753466
norm(A3*A3) = 0.795547
norm(A3*A4) = 0.278107
norm(A4*A1) = 0.832049
norm(A4*A2) = 0.474458
norm(A4*A3) = 0.228314
norm(A4*A4) = 0.018116
```

There are sixteen such pairs, and it turns out the maximum gain (matrix norm) of all such pairs is less than one. In fact, they are all less than 0.91. This means that for any $x(t)$, and any values of $A(t)$ and $A(t+1)$, we have

$$\|x(t+2)\| = \|A(t+1)A(t)x(t)\| \leq \|A(t+1)A(t)\|\|x(t)\| \leq 0.91\|x(t)\|.$$

Thus, in any *two steps*, the norm of $x(t)$ does indeed decrease! This implies that

$$\|x(2k)\| \leq 0.91^k\|x(0)\|.$$

We also have

$$\|x(2k+1)\| \leq 6.02\|x(2k)\| \leq (6.02)0.91^k\|x(0)\|.$$

Therefore we have $\|x(t)\| \to 0$ as $t \to \infty$, which means $x(t) \to 0$ as $t \to \infty$.

We mention one other valid approach that a few people used. We have already seen that the norms of 3 of the 4 matrices exceed one, which means that in any given step, the norm of the state can increase. This approach is based on changing coordinates so that, in the new coordinates, the norm of the state decreases at every step.

Suppose we can find a nonsingular matrix T for which $\|T^{-1}A_iT\| < 1$ for $i = 1, \ldots, 4$. Then the original system is stable, *i.e.*, all trajectories converge to zero. To see this, we note that

$$\begin{aligned} \|T^{-1}x(t)\| &= \|T^{-1}A(t-1)\cdots A(0)x(0)\| \\ &= \|T^{-1}A(t-1)T\cdots T^{-1}A(0)TT^{-1}x(0)\| \\ &\leq \alpha^t\|T^{-1}x(0)\|, \end{aligned}$$

where $\alpha = \max_i \|T^{-1}A_iT\| < 1$. What this shows is that at every step, $\|T^{-1}x\|$ decreases.

Now, how do you find such a T? This has a really good answer, which is well beyond EE263 (it's a convex problem, and can be solved using semidefinite programming and linear matrix inequalities). A few people found a valid T using random or other searches. As long as the method was *clearly* described, we can gave full credit for this approach.

We mention a few methods that are *wrong*. As mentioned above, any method that relies on eigenvalues, of any martix or matrices, is wrong.

One hand-waving method observed that the maximum gain output directions of each matrix didn't line up with the maximum gain input directions of any other matrix. (We saw several variations on this basic theme.) Well, that may be true, but we don't see what it proves.

Others tried to 'prove' that $x(t)$ goes to zero by simulating. Of course, that's not a proof. By the way, we wouldn't ask you to prove something that's false (at least not intentionally), so the fact that $x(t)$ converges to zero wasn't in question. We asked you to *prove* it.

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.

   Solution: The solution is

   $$(a)\ \sqrt{7^2+2^2} = 7.28, \qquad (b)\ 7, \qquad (c)\ 1, \qquad (d)\ 0.$$

   We didn't ask you to justify your answers, but of course we will. We start by writing

   $$\left\| [A\ B] \begin{bmatrix} u \\ v \end{bmatrix} \right\| = \|Au + Bv\|,$$

   where $u,\ v \in \mathbf{R}^3$.

   (a) First we note that

   $$\left\| [A\ B] \begin{bmatrix} u \\ v \end{bmatrix} \right\| = \|Au + Bv\| \leq \|Au\| + \|Bv\| \leq 7\|u\| + 2\|v\|.$$

   Now we ask: how large can $7\|u\| + 2\|v\|$ be, over all u and v with

   $$\left\| \begin{bmatrix} u \\ v \end{bmatrix} \right\| = \sqrt{\|u\|^2 + \|v\|^2} \leq 1.$$

   This is just Cauchy-Schwarz: it can be no larger than $\sqrt{7^2+2^2}$. Thus, for any matrices A and B with the given singular values, we have $\|C\| \leq \sqrt{7^2+2^2}$. Could

$\|C\|$ actually be this large? Yes: take

$$A = \begin{bmatrix} 7 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad u = \frac{1}{\sqrt{7^2 + 2^2}} \begin{bmatrix} 7 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \end{bmatrix}.$$

We have $\|C[u^T \; v^T]^T\| = \sqrt{7^2 + 2^2}$.

(b) Taking u to be the right singular vector of A associated with its largest singular value (*i.e.*, 7), and v zero, we find that

$$\left\| \begin{bmatrix} u \\ 0 \end{bmatrix} \right\| = 1, \qquad \left\| [A \; B] \begin{bmatrix} u \\ 0 \end{bmatrix} \right\| = \|Au\| = 7.$$

This shows that the norm of C must be at least 7. Can it be 7? The answer is yes: just take

$$A = \begin{bmatrix} 7 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

These matrices have the required singular values, and C is diagonal, evidently with $\|C\| = 7$. So the answer to part (b) is 7.

(c) Taking u zero and v to be the right singular vector of B associated with its smallest singular value (*i.e.*, 1), we have $\|C[u^T \; v^T]^T\| = 1$. The minimum gain of C could only be smaller, so we conclude that for any A and B we have $\sigma_6 \leq 1$. Could we attain this value for some choice of A and B? Yes:

$$A = \begin{bmatrix} 7 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

For these A and B, C is diagonal and has $\sigma_6 = 1$.

(d) The answer 0 is justified by

$$A = \begin{bmatrix} 7 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

for which $\sigma_6 = 0$.

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

To formalize these requirements, we use the objective $(1/\sqrt{n})\|T(t^{\mathrm{des}}) - T^{\mathrm{des}}\|$ and the constraint $(1/\sqrt{n})\|T(0)\| \leq T^{\mathrm{max}}$. The first expression is the RMS temperature deviation, at $t = t^{\mathrm{des}}$, 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 $T(0)$, $T(t^{\mathrm{des}})$ and T^{des}. Give the RMS temperature error $(1/\sqrt{n})\|T(t^{\mathrm{des}}) - T^{\mathrm{des}}\|$, and the RMS value of initial temperature $(1/\sqrt{n})\|T(0)\|$.

Solution.

(a) We can express the temperature dynamics as $\dot{T} = AT$, where A is a tridiagonal matrix with

$$A_{11} = -1/c_1(a_1 + b_1)$$

$$\begin{aligned} A_{ii} &= -1/c_i(a_i + b_i + b_{i-1}), \quad i = 2, \ldots, n, \\ A_{i,i-1} &= b_{i-1}/c_i, \quad i = 2, \ldots, n, \\ A_{i,i+1} &= b_i/c_i, \quad i = 1, \ldots, n-1. \end{aligned}$$

We have $T(t^{\mathrm{des}}) = e^{t^{\mathrm{des}}A}T(0)$. Therefore we must solve the problem

$$\begin{array}{ll} \text{minimize} & (1/n)\left\|e^{t^{\mathrm{des}}A}T(0) - T^{\mathrm{des}}\right\|^2 \\ \text{subject to} & (1/n)\|T(0)\|^2 \leq (T^{\mathrm{max}})^2. \end{array}$$

We solve this by minimizing

$$\left\|e^{t^{\mathrm{des}}A}T(0) - T^{\mathrm{des}}\right\|^2 + \rho\|T(0)\|^2,$$

and increasing ρ until $(1/n)\|T(0)\|^2 \leq (T^{\mathrm{max}})^2$ first holds (which will be with equality).

We mention one rather common error: simply obtaining the least-squares solution (without regards for $\|T(0)\|$, and then scaling this solution down so that the constraint is satisfied. This method produces results that look pretty good, when plotted, but are in fact not particularly good (in addition to just being wrong). This results in an RMS temperature error that more than 60% higher than using the correct method.

(b) The following code solves the problem in part (b).

```
clear all;  close all

temp_prof_data

A(1,1)=-1/c(1)*(a(1)+b(1));
A(1,2)=b(1)/c(1);
A(n,n)=-1/c(n)*(a(n)+b(n-1));
A(n,n-1)=b(n-1)/c(n);
for i=2:n-1
    A(i,i)=-1/c(i)*(a(i)+b(i)+b(i-1));
    A(i,i-1)=b(i-1)/c(i);
    A(i,i+1)=b(i)/c(i);
end


B=expm(A*tdes);
rho=0;
while 1
    C=[B; sqrt(rho)*eye(n)];
    d=[Tdes;zeros(n,1)];
```

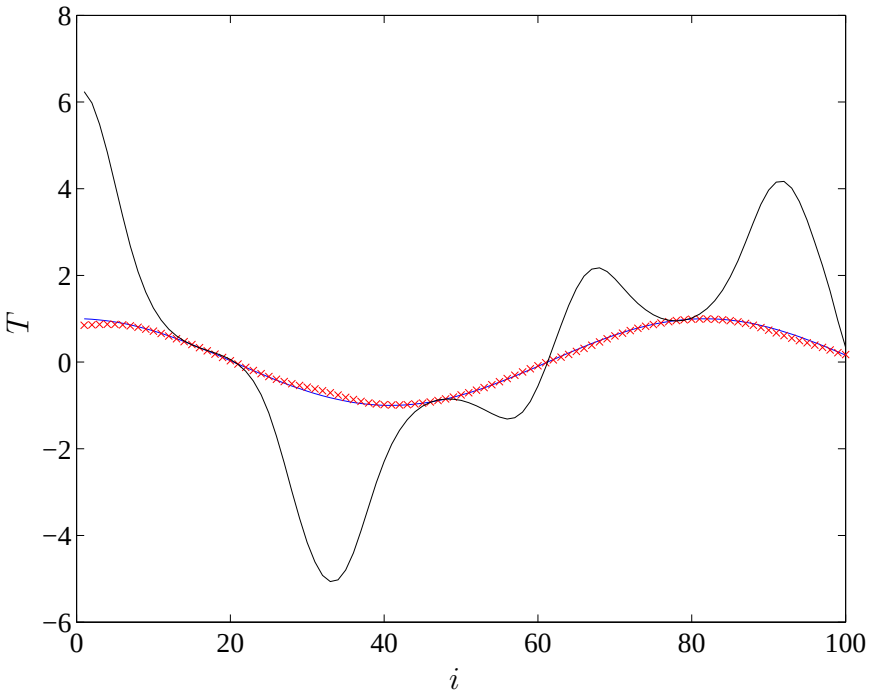


Figure 1: T^{des} (solid blue), $T(t^{\mathrm{des}})$ (dashed red) and $T(0)$ (dash-dotted black).

```
    T=C\d;
    if norm(T)/sqrt(n)<=Tmax
        break
    end
    rho=rho+1e-5;
end

plot(Tdes); hold on
plot(B*T,'xr');
plot(T,'k'); hold off
set(gca,'Fontname','Times','Fontsize', 16);
ylabel('t'); xlabel('x')
print -depsc temp_prof

norm(T)/sqrt(n)
norm(Tdes-B*T)/sqrt(n)
```

Figure 1 shows T^{des} with a solid blue line, $T(t^{\mathrm{des}})$ with a dashed red line and $T(0)$ with a dash-dotted black line. We have $(1/\sqrt{n})\|T(t^{\mathrm{des}}) - T^{\mathrm{des}}\| = 0.0457$, and, as expected, $(1/\sqrt{n})\|T(0)\| = 2.50$.

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.

Solution:

(a) Let c be the cut size encoded by x. The trick is to notice that

$$x^T A x = \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i x_j.$$

Whenever nodes i and j are connected by an edge, we get two contributions to the sum, *i.e.*, $x_i x_j$ and $x_j x_i$; when i and j are not connected by an edge, we get zero, since $A_{ij} = 0$. Thus the quadratic form above is equal to twice the sum, over all edges, of $x_i x_j$. But $x_i x_j = 1$ when nodes i and j are in the same group, and $x_i x_j = -1$ when nodes i and j are in different groups. If we let c be the number of edges between different groups, we have

$$x^T A x = 2((m - c) - c),$$

where $m - c$ is the number of edges inside groups and c is the number of edges between different groups. Solving for c we get

$$c = m/2 - x^T A x/4.$$

In other words, $P = -A/4$ and $d = m/2$. (We also have that $m = \mathbf{1}^T A \mathbf{1}/2$.)

(b) We seek a vector x that solves the problem

$$\begin{array}{ll} \text{maximize} & x^T P x \\ \text{subject to} & x^T x = n. \end{array}$$

Let $y = x/\sqrt{n}$ and the problem becomes

$$\begin{array}{ll} \text{maximize} & n y^T P y \\ \text{subject to} & y^T y = 1. \end{array}$$

The optimal value of this problem is $n\lambda_{\max}$, where $\lambda_{\max}$ is the largest eigenvalue of P. It is achieved by $y = v_{\max}$, where $v_{\max}$ is normalized eigenvector of P associated with the maximum eigenvalue. Therefore we can take $x^\star = \sqrt{n} v_{\max}$.

(c) The following script implements the heuristic and compares its cut size to the cut sizes obtained from random partitions.

```
mc_data;

P = -A/4;
d = m/2;

% MAXCUT heuristic
[V,D] = eig(P);
idx = find(diag(D)==max(diag(D)));
x = sign(V(:,idx));
c = x'*P*x + d;
disp('The cut size returned by the heuristic is:');
disp(c);

% random cuts
c_rd = 0;
for iter = 1:1000
    temp = sign(rand(n,1)-0.5);
    ctemp = temp'*P*temp + d;
    if ctemp>c_rd
        x_rd = temp;
        c_rd = ctemp;
    end
end
disp('The best cut size over 1000 random partitions is:');
disp(c_rd);
```

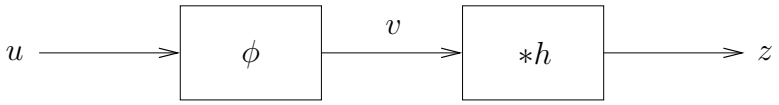
The script returns the following results.

```
The cut size returned by the heuristic is:
```

```
    69

The best cut size over 1000 random partitions is:
    51
```

The given graph has only $m = 72$ edges, so we have found a partition in which almost all edges go between the different groups.

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

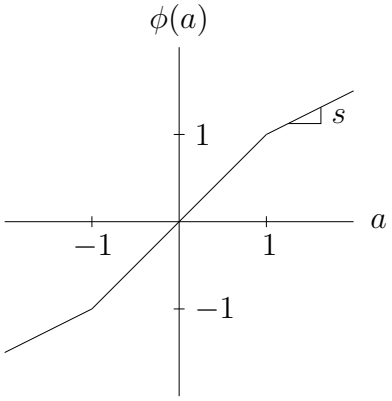
$$z(t) = \sum_{\tau=0}^{M-1} h(\tau)v(t-\tau), \qquad v(t) = \phi(u(t)), \qquad t \in \mathbf{Z}.$$

(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

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

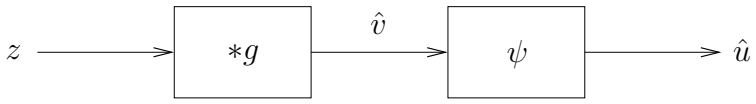
where $s > 0$ is a parameter. This function is shown below.



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 $t = 1, \ldots, N$.

Solution.

(a) The estimate $\hat{v}(i)$ is a linear function of g, and $\phi(u(i))$ is an affine function of s (*i.e.*, linear plus a constant). Thus, the quantity

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

can be written as $J = (1/(N-M+1))\|y - Ax\|^2$, where

$$x = [g(M-1) \ \cdots \ g(0) \ s]^T,$$

$$y_i = \begin{cases} 1 & u(i+M-1) > 1 \\ -1 & u(i+M-1) < -1 \\ u(i+M-1) & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, N-M+1$, and

$$A = \begin{bmatrix} z(1) & \cdots & z(M) & w_1 \\ \vdots & \ddots & \vdots & \vdots \\ z(N-M+1) & \cdots & z(N) & w_{N-M+1} \end{bmatrix},$$

where

$$w_i = \begin{cases} 1 - u(i+M-1) & u(i+M-1) > 1 \\ -1 - u(i+M-1) & u(i+M-1) < -1 \\ 0 & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, N-M+1$. Minimizing J is a least-squares problem and the solution is given by

$$x = (A^T A)^{-1} A^T y.$$

Several people used a method that gives very close to the correct solution, but is more complicated than it needs to be, so we took off 5 or so points (depending on the clarity). The method treats s as a parameter, not as a variable to be determined. For fixed s, we get a least-squares problem for g, which we solve. Then we have an out loop that sweeps over many values of s, to see which one has the smallest objective value.

(b) The following matlab code implements the solution.

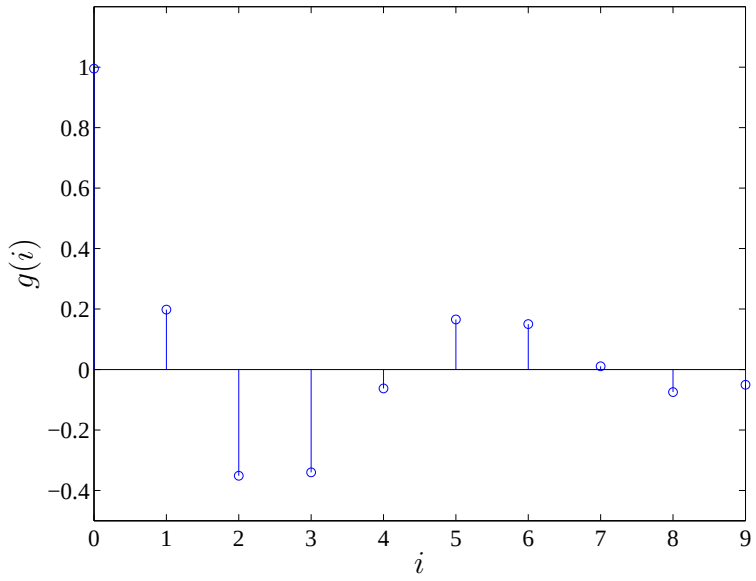
```
clear all
nleq_data;

% Forming least-squares problem
% x = [g(M-1) ... g(0) s]'
y = []; A = []; k = 0;
```

```
for i=M:N
    k = k+1;
    A(k,1:M) = z(i-M+1:i);
    if u(i)>1
        y(k,1) = 1;
        A(k,M+1) = 1 - u(i);
    elseif u(i) < -1
        y(k,1) = -1;
        A(k, M+1) = -1 - u(i);
    else
        y(k,1) =  u(i);
        A(k, M+1) = 0;
    end
end

x = A\y;
g = flipud(x(1:M));
s = x(end);
fprintf('\nSaturation gain estimate s hat = %f\n', s);
J = (1/(N-M+1))*((norm(y - A*x))^2);
fprintf('J = %e\n', J);

vhat =  conv(g,z);
uhat = [];
for i =1:length(vhat)
    if vhat(i)>1
        uhat(i,1) = (vhat(i)-1+s)/s;

    elseif vhat(i) < -1
        uhat(i,1) = (vhat(i)+1-s)/s;

    else
        uhat(i,1) = vhat(i);

    end
end
```

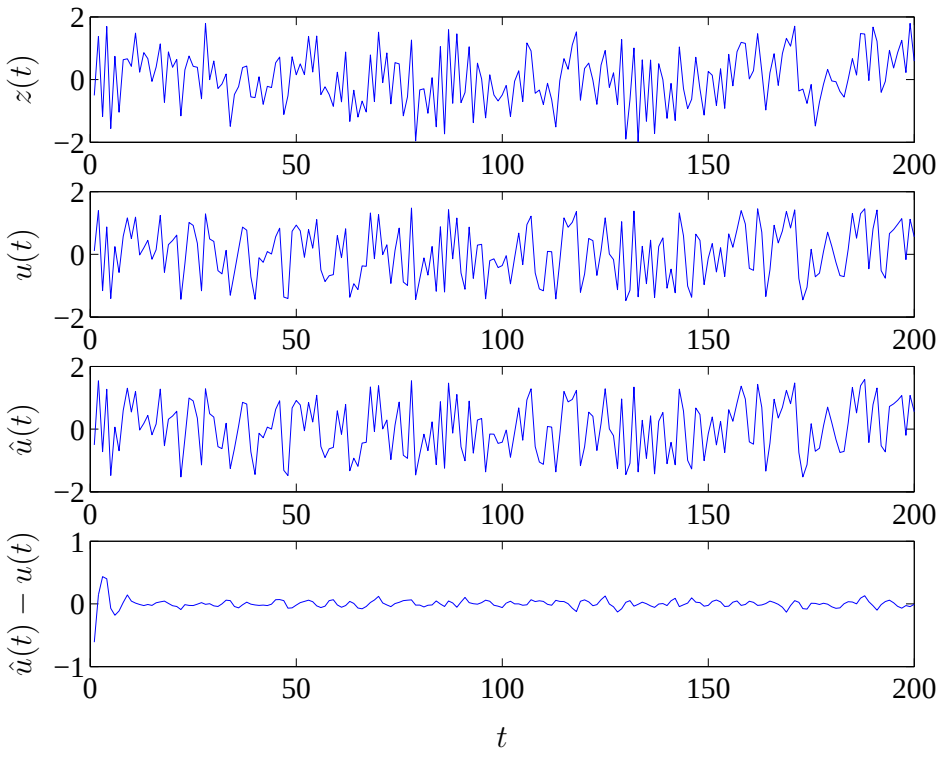
The solution obtained is:

```
g
0 0.994895
1 0.198037
2 -0.351337
3 -0.339969
```

```
4 -0.062389
5 0.165468
6 0.150345
7 0.010682
8 -0.074457
9 -0.050343

Saturation gain estimate s hat = 0.500388
J = 1.200482e-03
```



(c) The input signal $u(t)$, the output signal $z(t)$, the equalized signal $\hat{u}(t)$ (assuming $z(t) = 0$, $t \leq 0$), and the equalization error $\hat{u}(t) - u(t)$, for $t = 1, \ldots, N$ are shown below.


$z(t)$
$u(t)$
$\hat{u}(t)$
$\hat{u}(t) - u(t)$
t
2
0
−2
1
−1
0
50
100
150
200

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.)

Solution.

(a) We have

$$\begin{aligned}
\frac{d}{dt}\|x(t)\|^2 + \rho\|u(t)\|^2 &= \frac{d}{dt}(x(t)^T x(t)) + \rho\|u(t)\|^2 \\
&= \dot{x}(t)^T x(t) + x(t)^T \dot{x}(t) + \rho\|u(t)\|^2 \\
&= (Ax(t) + Bu(t))^T x(t) + x(t)^T (Ax(t) + Bu(t)) + \rho\|u(t)\|^2 \\
&= x(t)^T (A^T + A)x(t) + 2x(t)^T Bu(t) + \rho\|u(t)\|^2.
\end{aligned}$$

To minimize this with respect to $u(t)$, we set the gradient equal to zero:

$$2B^T x(t) + 2\rho u(t) = 0.$$

This yields $u(t) = Kx(t)$, with $K = -(1/\rho)B^T$.

For use in part (c), we note that the closed-loop dynamics are given by

$$\dot{x} = Ax + Bu = (A - (1/\rho)BB^T)x.$$

(b) With this value of $u(t)$, we have

$$\frac{d}{dt}\|x(t)\|^2 = x(t)^T \left(A^T + A - (2/\rho)BB^T\right) x(t).$$

This is negative for all nonzero $x(t)$ if and only if the matrix in the quadratic form is negative definite, *i.e.*,

$$A^T + A - (2/\rho)BB^T \prec 0.$$

There are many other ways to write this condition.

(c) We can't find an example with $n = 1$, but for $n = 2$ we can use

$$A = \begin{bmatrix} -0.1 & 1 \\ 0 & -0.1 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \rho = 1.$$

Clearly A has negative eigenvalues and therefore the open-loop is stable but as the following Matlab code shows, the closed-loop system is unstable. The Matlab code is

```
A=[-0.1  1; 0 -0.1];
B=[1 -1]';
rho=1;

eig(A)
eig(A-(1/rho)*B*B')
```

The output of the above code is

```
 ans =
   -0.1000
   -0.1000
ans =
    0.3142
   -2.5142.
```