EE364a Convex Optimization I Prof. S. Boyd
March 14–15 or March 15–16, 2008.

Final exam solutions

You may use any books, notes, or computer programs (*e.g.*, Matlab, `cvx`), but you may not discuss the exam with anyone until March 18, after everyone has taken the exam. The only exception is that you can ask us for clarification, via the course staff email address. We've tried pretty hard to make the exam unambiguous and clear, so we're unlikely to say much.

Please make a copy of your exam before handing it in.

Please attach the cover page to the front of your exam. Assemble your solutions in order (problem 1, problem 2, problem 3, ...), starting a new page for each problem. Put everything associated with each problem (*e.g.*, text, code, plots) together; do not attach code of plots at the end of the final.

We will deduct points from long needlessly complex solutions, even if they are correct. Our solutions are not long, so if you find that your solution to a problem goes on and on for many pages, you should try to figure out a simpler one. We expect neat, legible exams from everyone, including those enrolled Cr/N.

When a problem involves computation you must give all of the following: a clear discussion and justification of exactly what you did, the Matlab (or other) source code that produces the result, and the final numerical results or plots.
To download Matlab files containing problem data, you'll have to type the whole URL given in the problem into your browser; there are no links on the course web page pointing to these files. To get a file called `filename.m`, for example, you would retrieve

```
http://www.stanford.edu/class/ee364a/final-data/filename.m
```

with your browser.

Please respect the honor code. Although we allow you to work on homework assignments in small groups, you cannot discuss the final with anyone, at least until everyone has taken it.

All problems have equal weight. Some are easier than they might appear at first glance.

Download the most recent version of `cvx`.

Be sure to check your email often during the exam, just in case we need to send out an important announcement.

1. *Optimal investment to fund an expense stream.* An organization (such as a municipality) knows its operating expenses over the next T periods, denoted $E_1, \ldots, E_T$. (Normally these are positive; but we can have negative E_t, which corresponds to income.) These expenses will be funded by a combination of investment income, from a mixture of bonds purchased at $t = 0$, and a cash account.

   The bonds generate investment income, denoted $I_1, \ldots, I_T$. The cash balance is denoted $B_0, \ldots, B_T$, where $B_0 \geq 0$ is the amount of the initial deposit into the cash account. We can have $B_t < 0$ for $t = 1, \ldots, T$, which represents borrowing.

   After paying for the expenses using investment income and cash, in period t, we are left with $B_t - E_t + I_t$ in cash. If this amount is positive, it earns interest at the rate $r_+ > 0$; if it is negative, we must pay interest at rate r_-, where $r_- \geq r_+$. Thus the expenses, investment income, and cash balances are linked as follows:

$$B_{t+1} = \begin{cases} (1 + r_+)(B_t - E_t + I_t) & B_t - E_t + I_t \geq 0 \\ (1 + r_-)(B_t - E_t + I_t) & B_t - E_t + I_t < 0, \end{cases}$$

   for $t = 1, \ldots, T - 1$. We take $B_1 = (1 + r_+)B_0$, and we require that $B_T - E_T + I_T = 0$, which means the final cash balance, plus income, exactly covers the final expense.

   The initial investment will be a mixture of bonds, labeled $1, \ldots, n$. Bond i has a price $P_i > 0$, a coupon payment $C_i > 0$, and a maturity $M_i \in \{1, \ldots, T\}$. Bond i generates an income stream given by

$$a_t^{(i)} = \begin{cases} C_i & t < M_i \\ C_i + 1 & t = M_i \\ 0 & t > M_i, \end{cases}$$

   for $t = 1, \ldots, T$. If x_i is the number of units of bond i purchased (at $t = 0$), the total investment cash flow is

$$I_t = x_1 a_t^{(1)} + \cdots + x_n a_t^{(n)}, \quad t = 1, \ldots, T.$$

   We will require $x_i \geq 0$. (The x_i can be fractional; they do not need to be integers.)

   The total initial investment required to purchase the bonds, and fund the initial cash balance at $t = 0$, is $x_1 P_1 + \cdots + x_n P_n + B_0$.

   (a) Explain how to choose x and B_0 to minimize the total initial investment required to fund the expense stream.

   (b) Solve the problem instance given in `opt_funding_data.m`. Give optimal values of x and B_0. Give the optimal total initial investment, and compare it to the initial investment required if no bonds were purchased (which would mean that all the expenses were funded from the cash account). Plot the cash balance (versus period) with optimal bond investment, and with no bond investment.

   Solution.

   (a) We will give several solutions (which are close, but not the same).

Method 1. The cash balance propagation equations,

$$B_{t+1} = \begin{cases} (1+r_+)(B_t - E_t + I_t) & B_t - E_t + I_t \geq 0 \\ (1+r_-)(B_t - E_t + I_t) & B_t - E_t + I_t < 0, \end{cases}$$

which can be expressed as

$$B_{t+1} = \min\left\{(1+r_+)(B_t - E_t + I_t), (1+r_-)(B_t - E_t + I_t)\right\},$$

are clearly not convex when B_t are considered as variables. We will describe two ways to handle them.

In the first method, we relax the propagation equations to the convex inequalities

$$B_{t+1} \leq \min\left\{(1+r_+)(B_t - E_t + I_t), (1+r_-)(B_t - E_t + I_t)\right\}.$$

With this relaxation, the problem can be expressed as:

$$\begin{array}{ll} \text{minimize} & B_0 + \sum_{i=1}^n P_i x_i \\ \text{subject to} & \sum_{i=1}^n a_t^{(i)} x_i = I_t, \quad t = 1, \ldots, T \\ & x \succeq 0, \quad B_0 \geq 0 \\ & B_1 = (1+r_+)B_0 \\ & B_{t+1} \leq \min\left\{(1+r_+)(B_t - E_t + I_t), (1+r_-)(B_t - E_t + I_t)\right\}, \ t = 1, \ldots, T-1 \\ & B_T - E_T + I_T = 0, \end{array}$$

with variables $B_0, \ldots, B_T$, $I_1, \ldots, I_T$, $x_1, \ldots, x_n$. (We could treat I_t as an affine expression, instead of a variable; it makes no difference.)

We will now argue that any solution of this problem must satisfy

$$B_{t+1} = \min\left\{(1+r_+)(B_t - E_t + I_t), (1+r_-)(B_t - E_t + I_t)\right\}, \quad t = 1, \ldots, T-1,$$

which means that by solving the LP above, we are also solving the original problem. Suppose this is not the case; suppose, for example, that τ is the first period for which

$$B_{\tau+1} < \min\left\{(1+r_+)(B_\tau - E_\tau + I_\tau), (1+r_-)(B_\tau - E_\tau + I_\tau)\right\}.$$

Of course, this is kind of silly: it means we have elected to throw away some cash between periods τ and $\tau+1$. To give a formal argument that this cannot happen, we first note that B_{t+1} is monotonically nondecreasing in B_t for all t, since each term within the mininimization is monotonically increasing in B_t. Thus, we can decrease B_τ until $B_{\tau+1} = \min\left\{(1+r_+)(B_\tau - E_\tau + I_\tau), (1+r_-)(B_\tau - E_\tau + I_\tau)\right\}$, and still maintain feasibility of the cash stream. By induction, we can decrease $B_{\tau-1}$, $B_{\tau-2}$, $\ldots$, B_0 and still have a feasible solution. This new stream is feasible, and reduces B_0, which shows that the original point was not optimal.

Method 2. The second method carries out a different relaxation. In this method we do not consider $B_1, \ldots, B_T$ as variables; instead we think of them as functions of x and B_0, defined by a recursion. We will show that $B_T(B_0, x)$ is a concave function of B_0 and x. $B_1(B_0, x) = (1 + r_+)B_0$ is an concave (affine) function. If $B_t(B_0, x)$ is a concave function, then

$$\begin{aligned} B_{t+1}(B_0, x) &= \min\{(1 + r_+)(B_t(B_0, x) - E_t + I_t(x)), \\ &\qquad (1 + r_-)(B_t(B_0, x) - E_t + I_t(x))\}, \quad t = 1, \ldots, T - 1 \end{aligned}$$

is also a concave function because it is the pointwise minimum of two concave functions. Therefore, we can conclude that $B_T(B_0, x)$ is a concave function.

The final cash balance constraint $B_T(B_0, x) - E_T + I_T(x) = 0$ is clearly not convex because $B_T(B_0, x)$ is a concave function. To handle this equation, we relax it to the convex inequality

$$B_T(B_0, x) - E_T + I_T(x) \geq 0.$$

With this relaxation, the problem can be expressed as:

$$\begin{array}{ll} \text{minimize} & B_0 + \sum_{i=1}^n P_i x_i \\ \text{subject to} & x \succeq 0, \quad B_0 \geq 0 \\ & B_T(B_0, x) - E_T + I_T(x) \geq 0, \end{array}$$

with variables B_0 and $x_1, \ldots, x_n$. Here $B_T(B_0, x)$ and $I_T(x)$ are defined by the following recursive definitions:

$$\begin{aligned} B_1(B_0, x) &= (1 + r_+)B_0 \\ B_{t+1}(B_0, x) &= \min\{(1 + r_+)(B_t(B_0, x) - E_t + I_t(x)), \\ &\qquad (1 + r_-)(B_t(B_0, x) - E_t + I_t(x))\}, \quad t = 1, \ldots, T - 1 \\ I_t(x) &= \sum_{i=1}^n a_t^{(i)} x_i, \quad t = 1, \ldots, T. \end{aligned}$$

We can argue that any solution of this problem must satisfy

$$B_T(B_0, x) - E_T + I_T(x) = 0,$$

which is very similar to the argument given above (for the inequality relaxation).

Method 3. Here is yet another solution to this problem. The cash balance propagation equations,

$$B_{t+1} = \begin{cases} (1 + r_+)(B_t - E_t + I_t) & B_t - E_t + I_t \geq 0 \\ (1 + r_-)(B_t - E_t + I_t) & B_t - E_t + I_t < 0, \end{cases}$$

are equivalent to LP constraints

$$\begin{aligned} B_{t+1} &= (1+r_+)S_t^+ - (1+r_-)S_t^-, \\ B_t - E_t + I_t &= S_t^+ - S_t^-, \\ S_t^+ &\geq 0, \\ S_t^- &\geq 0. \end{aligned}$$

The equivalence can be shown easily: Suppose $B_t - E_t + I_t$ is positive, then $S_t^+ = B_t - E_t + I_t$ and $S_t^- = 0$, so, $B_{t+1} = (1+r_+)(B_t - E_t + I_t)$. If $B_t - E_t + I_t$ is negative, then $S_t^+ = 0$ and $S_t^- = B_t - E_t + I_t$, so, $B_{t+1} = (1+r_-)(B_t - E_t + I_t)$. Therefore, the problem can be expressed as a LP:

$$\begin{array}{ll} \text{minimize} & B_0 + \sum_{i=1}^n P_i x_i \\ \text{subject to} & \sum_{i=1}^n a_t^{(i)} x_i = I_t, \quad t = 1, \ldots, T \\ & x \succeq 0, \quad B_0 \geq 0 \\ & B_1 = (1+r_+)B_0 \\ & B_T - E_T + I_T = 0, \\ & B_{t+1} = (1+r_+)S_t^+ - (1+r_-)S_t^-, \quad t = 1, \ldots, T-1 \\ & B_t - E_t + I_t = S_t^+ - S_t^-, \quad t = 1, \ldots, T-1 \\ & S_t^+ \geq 0, \quad t = 1, \ldots, T-1 \\ & S_t^- \geq 0 \quad t = 1, \ldots, T-1, \end{array}$$

with variables $B_0, \ldots, B_T$, $I_1, \ldots, I_T$, $x_1, \ldots, x_n$, $S_1^+, \ldots, S_{T-1}^+$ and $S_1^+, \ldots, S_{T-1}^-$. With this approach, we have to argue that for any t, we will either have S_t^+ or S_t^- zero. (This is easily seen.)

(b)
```
% Optimal investment to fund an expense stream
clear all; close all;
opt_funding_data;

cvx_begin
variables x(n) B0 B(T) I(T)
minimize( P'*x+B0 )
subject to
    I == A*x
    x >= 0 % use x == 0 for no bond purchase
    B0 >= 0
    B(1) == (1+rp)*B0
    for t = 1:T-1
        B(t+1) <= min( (1+rp)*(B(t)-E(t)+I(t)), (1+rn)*(B(t)-E(t)+I(t)) )
    end
    B(T)-E(T)+I(T) == 0
cvx_end
```

```
% optimal total initial investment
cvx_optval

% associated optimal cash balance plot
bar(0:T,[B0;B]);
ylabel('cash balance'); xlabel('t');
```

The code (cvx part) with recursive definition of $B_t(B_0, x)$ is given below.

```
cvx_begin
variables x(n) B0
minimize( P'*x+B0 )
subject to
    x >= 0 % use x == 0 for no bond purchase
    B0 >= 0
    B(1) = (1+rp)*B0
    for t = 1:T-1
        B(t+1)=min((1+rp)*(B(t)-E(t)+A(t,:)*x),(1+rn)*(B(t)-E(t)+A(t,:)*x));
    end
    B(T)-E(T)+A(T,:)*x >= 0
cvx_end
```

The code (cvx part) with LP formulation is given below.

```
cvx_begin
variables x(n) B0 B(T) I(T) Sp(T-1) Sn(T-1)
minimize( P'*x+B0 )
subject to
    I == A*x
    x >= 0 % use x == 0 for no bond purchase
    B0 >= 0
    B(1) == (1+rp)*B0
    Sp >= 0
    Sn >= 0
    for t = 1:T-1
        B(t)-E(t)+I(t) == Sp(t) - Sn(t);
        B(t+1) == (1+rp)*Sp(t)-(1+rn)*Sn(t)
    end
    B(T)-E(T)+I(T) == 0
cvx_end
```

The optimal bond investment is

$$x = (0.0000, 18.9326, 0.0000, 0.0000, 13.8493, 8.9228),$$

and the optimal initial deposit to the cash account is $B_0 = 0$. The optimal total initial investment is 40.7495.

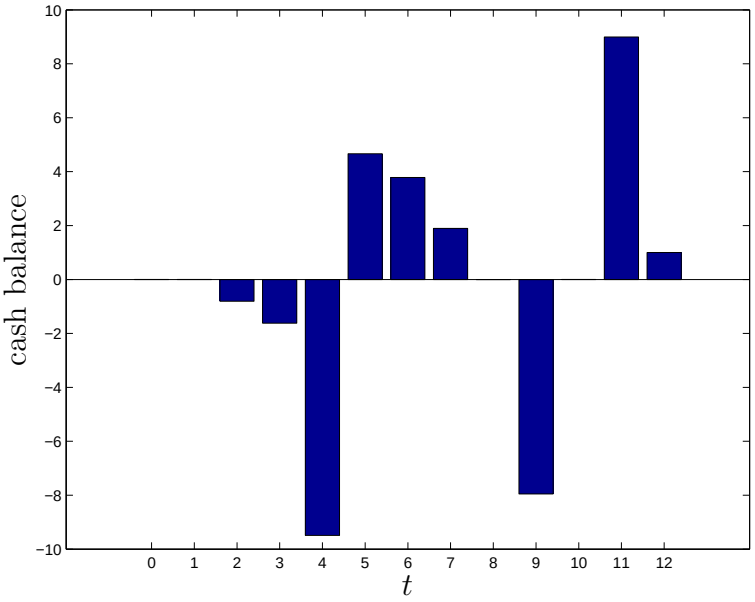


Figure 1: Cash balance with optimal bond purchase.

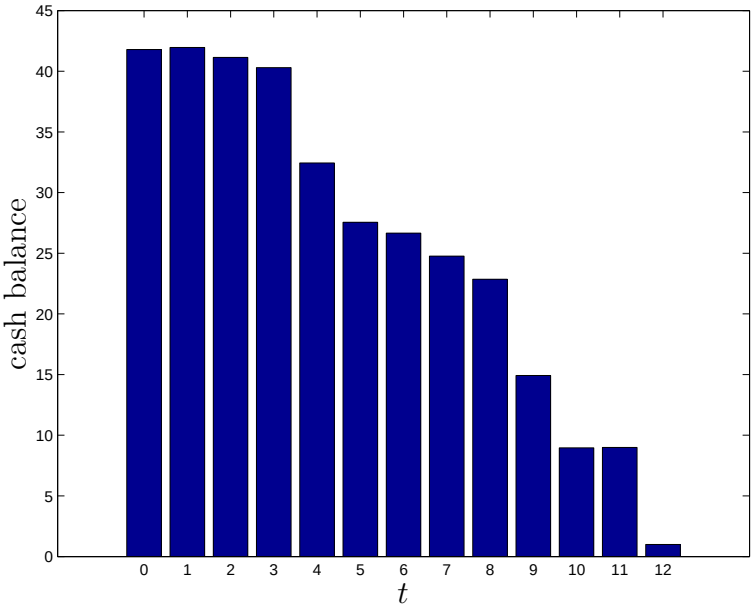


Figure 2: Cash balance with no bond purchase.

With no bond investment (*i.e.*, $x = 0$), an initial deposit of $B_0 = 41.7902$ is cash is required, around 2.5% more than when we invest in bonds.

2. *Utility versus latency trade-off in a network.* We consider a network with m edges, labeled $1, \ldots, m$, and n flows, labeled $1, \ldots, n$. Each flow has an associated nonnegative flow rate f_j; each edge or link has an associated positive capacity c_i. Each flow passes over a fixed set of links (its route); the total traffic t_i on link i is the sum of the flow rates over all flows that pass through link i. The flow routes are described by a routing matrix $R \in \mathbf{R}^{m \times n}$, defined as

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

Thus, the vector of link traffic, $t \in \mathbf{R}^m$, is given by $t = Rf$. The link capacity constraint can be expressed as $Rf \preceq c$. The (logarithmic) network utility is defined as $U(f) = \sum_{j=1}^{n} \log f_j$.

The (average queuing) delay on link i is given by

$$d_i = \frac{1}{c_i - t_i}$$

(multiplied by a constant, that doesn't matter to us). We take $d_i = \infty$ for $t_i = c_i$. The delay or latency for flow j, denoted l_j, is the sum of the link delays over all links that flow j passes through. We define the maximum flow latency as

$$L = \max\{l_1, \ldots, l_n\}.$$

We are given R and c; we are to choose f.

(a) How would you find the flow rates that maximize the utility U, ignoring flow latency? (In particular, we allow $L = \infty$.) We'll refer to this maximum achievable utility as $U^{\max}$.

(b) How would you find the flow rates that minimize the maximum flow latency L, ignoring utility? (In particular, we allow $U = -\infty$.) We'll refer to this minimum achievable latency as $L^{\min}$.

(c) Explain how to find the optimal trade-off between utility U (which we want to maximize) and latency L (which we want to minimize).

(d) Find $U^{\max}$, $L^{\min}$, and plot the optimal trade-off of utility versus latency for the network with data given in `net_util_data.m`, showing $L^{\min}$ and $U^{\max}$ on the same plot. Your plot should cover the range from $L = 1.1L^{\min}$ to $L = 11L^{\min}$. Plot U vertically, on a linear scale, and L horizontally, using a log scale.

Note. For parts (a), (b), and (c), your answer can involve solving one or more convex optimization problems. But if there is a simpler solution, you should say so.

Solution.

(a) To maximize utility we solve the convex problem

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^{n} \log f_j \\ \text{subject to} & Rf \preceq c. \end{array}$$

with variable f, and the implicit constraint $f \succeq 0$.

(b) Link delay is monotonically increasing in traffic, so it is minimized with zero traffic. Since flow latency is the sum of link delays, it is also minimized by the choice $f = 0$. So zero flow minimizes each flow latency. With $f = 0$ we have $d_i = 1/c_i$. To sum these delays over the routes, we multiply by R^T to get $l = R^T(1/c_1, \ldots, 1/c_m)$, where l is the vector of flow latencies. Thus we have

$$L^{\min} = \max\left(R^T(1/c_1, \ldots, 1/c_m)\right),$$

where the max is the maximum over the entries of the vector $R^T(1/c_1, \ldots, 1/c_m)$.

(c) Let $b_1^T, \ldots, b_m^T$ denote the rows of R. To find the optimal trade-off between U and L, we solve the problem

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^{n} \log f_j \\ \text{subject to} & Rf \preceq c, \\ & \displaystyle\sum_{i=1}^{m} \frac{R_{ij}}{c_i - b_i^T f} \leq L, \quad j = 1, \ldots, n, \end{array}$$

for a range of values of L. The variable here is f, and there is an implicit constraint $f \succeq 0$. Evidently, if $Rf \preceq c$ we must have $c_i - b_i^T f \geq 0$, so the constraints

$$\sum_{i=1}^{m} \frac{R_{ij}}{c_i - b_i^T f} \leq L, \quad j = 1, \ldots, n,$$

are convex. This is therefore a convex optimization problem.

(d) The following code computes the utility versus latency trade-off.

```
% solution for network utility problem
clear all;
net_util_data;

% let's find max utility with no delay constraint
cvx_begin
    variable f(n)
    maximize geomean(f)
    R*f <= c
    f >= 0; % not needed; enforced by geomean domain
cvx_end
Umax=sum(log(f));
```

```
% let's find min latency with no utility constraint
% can be done analytically: just take f=0
Lmin = max(R'*(1./c));

% now let's do pareto curve

N = 20;
ds = 1.10*Lmin*logspace(0,1,N);  % go from 10% above L
Uopt = [];

for d = ds
    cvx_begin
        variable f(n)
        maximize geomean(f);
        R'*inv_pos(c-R*f) <= d*ones(n,1)
        f >= 0; % not needed; enforced by geomean domain
        R*f <= c; % not needed; enforced by inv_pos domain
    cvx_end
    Uopt = [Uopt n*log(cvx_optval)];
end

semilogx(ds,Uopt,'k-',[Lmin,ds],[Umax,ones(1,N)*Umax],...
        'k--',[1,1]*Lmin,[Uopt(1),Umax],'k--')
% axis([ds(1), ds(N), -480, -340]);
xlabel('L'); ylabel('U');
```

The trade-off curve is shown in figure 3.

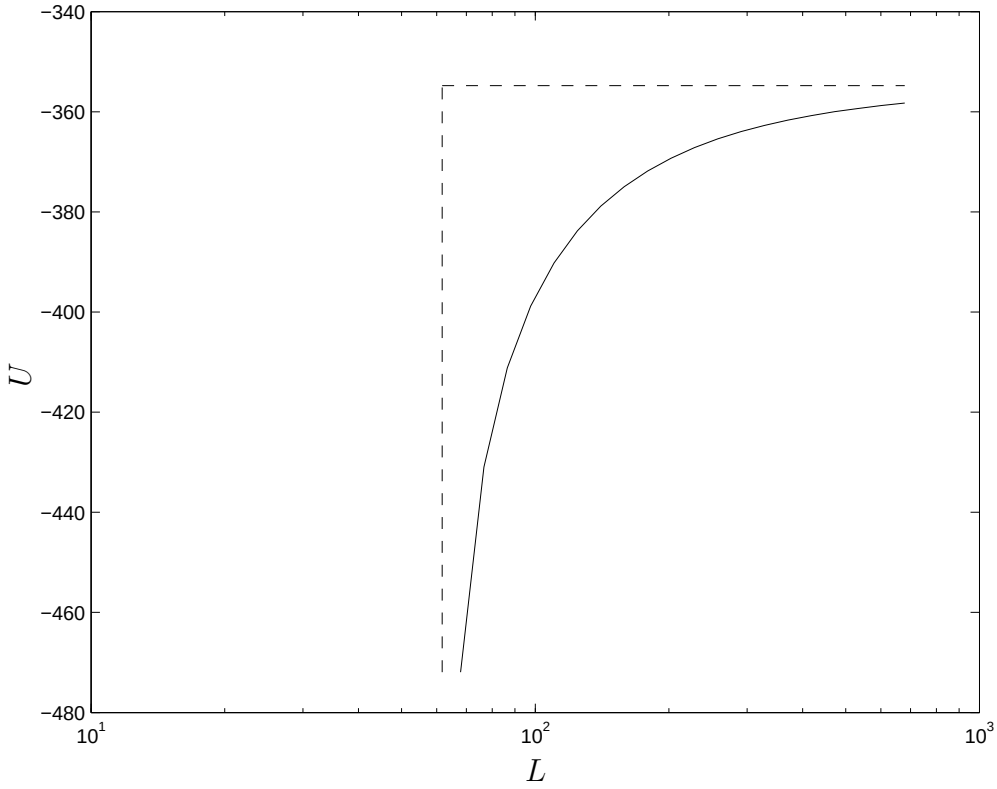


Figure 3: Utility versus maximum latency trade-off. Solid: trade-off curve, dashed: $L_{\min}$ and $U_{\max}$

3. *Optimal design of a tensile structure.* A tensile structure is modeled as a set of n masses in $\mathbf{R}^2$, some of which are fixed, connected by a set of N springs. The masses are in equilibrium, with spring forces, connection forces for the fixed masses, and gravity balanced. (This equilibrium occurs when the position of the masses minimizes the total energy, defined below.)

   We let $(x_i, y_i) \in \mathbf{R}^2$ denote the position of mass i, and $m_i > 0$ its mass value. The first p masses are fixed, which means that $x_i = x_i^{\text{fixed}}$ and $y_i = y_i^{\text{fixed}}$, for $i = 1, \ldots, p$. The gravitational potential energy of mass i is gm_iy_i, where $g \approx 9.8$ is the gravitational acceleration.

   Suppose spring j connects masses r and s. Its elastic potential energy is

$$(1/2)k_j \left((x_r - x_s)^2 + (y_r - y_s)^2 \right),$$

   where $k_j \geq 0$ is the stiffness of spring j.

   To describe the topology, *i.e.*, which springs connect which masses, we will use the incidence matrx $A \in \mathbf{R}^{n \times N}$, defined as

$$A_{ij} = \begin{cases} 1 & \text{head of spring } j \text{ connects to mass } i \\ -1 & \text{tail of spring } j \text{ connects to mass } i \\ 0 & \text{otherwise.} \end{cases}$$

   Here we arbitrarily choose a head and tail for each spring, but in fact the springs are completely symmetric, and the choice can be reversed without any effect. (Hopefully you will discover why it is convenient to use the incidence matrix A to specify the topology of the system.)

   The total energy is the sum of the gravitational energies, over all the masses, plus the sum of the elastic energies, over all springs. The equilibrium positions of the masses is the point that minimizes the total energy, subject to the constraints that the first p positions are fixed. (In the equilibrium positions, the total force on each mass is zero.) We let E_{min} denote the total energy of the system, in its equilibrium position. (We assume the energy is bounded below; this occurs if and only if each mass is connected, through some set of springs with positive stiffness, to a fixed mass.)

   The total energy E_{min} is a measure of the stiffness of the structure, with larger E_{min} corresponding to stiffer. (We can think of $E_{\text{min}} = -\infty$ as an infinitely unstiff structure; in this case, at least one mass is not even supported against gravity.)

   (a) Suppose we know the fixed positions $x_1^{\text{fixed}}, \ldots, x_p^{\text{fixed}}$, $y_1^{\text{fixed}}, \ldots, y_p^{\text{fixed}}$, the mass values $m_1, \ldots, m_n$, the spring topology A, and the constant g. You are to choose nonnegative $k_1, \ldots, k_N$, subject to a budget constraint $\mathbf{1}^T k = k_1 + \cdots + k_N = k^{\text{tot}}$, where k^{tot} is given. Your goal is to maximize E_{min}.

   Explain how to do this using convex optimization.

(b) Carry out your method for the problem data given in `tens_struct_data.m`. This file defines all the needed data, and also plots the equilibrium configuration when the stiffness is evenly distributed across the springs (*i.e.*, $k = (k^{\text{tot}}/N)\mathbf{1}$).

Report the optimal value of $E_{\min}$. Plot the optimized equilibrium configuration, and compare it to the equilibrium configuration with evenly distributed stiffness. (The code for doing this is in the file `tens_struct_data.m`, but commented out.)

Solution:

(a) $A^T x$ gives a vector of the x-displacements of the springs, and $A^T y$ gives a vector of the y-displacements of the springs.

The energy as a function of x, y, and k is given by

$$E(x, y, k) = (1/2)x^T A\,\mathbf{diag}(k)A^T x + (1/2)y^T A\,\mathbf{diag}(k)A^T y + c^T y,$$

where $c_i = gm_i$. This is an affine function of k. Thus the minimum energy is the minimum of this function, over all x and y that satisfy the fixed constraints. It follows immediately that $E_{\min}$ is a concave function of k. That's good, since we want to maximize it.

We'll need an explicit formula for $E_{\min}$. To do this we partition A into A_1 and A_2, where $A_1 \in \mathbf{R}^{p\times N}$ is made up of the first p rows of A, while $A_2 \in \mathbf{R}^{(n-p)\times N}$ is made up of the last $n-p$ rows of A.

Let $\bar{x}, \bar{y}, \bar{c} \in \mathbf{R}^{n-p}$ denote the last $n-p$ (*i.e.*, free) elements of x, y and c respectively. The minimum energy can be written as

$$E_{\min}(k) = \min_{\bar{x},\bar{y}} \left((1/2)z_x^T\,\mathbf{diag}(k)z_x + (1/2)z_y^T\,\mathbf{diag}(k)z_y + \bar{c}^T\bar{y} + C\right),$$

where $C = \sum_{i=1}^{p} gm_i y_i^{\text{fixed}}$ and

$$\begin{aligned} z_x &= A_2^T\bar{x} + b_x \\ z_y &= A_2^T\bar{y} + b_y \\ b_x &= A_1^T x^{\text{fixed}} \\ b_y &= A_1^T y^{\text{fixed}}. \end{aligned}$$

Thus to evaluate $E_{\min}$ we need to evaluate the minimum of an unconstrained quadratic in $\bar{x}$ and $\bar{y}$. This gives us

$$E_{\min}(k) = (1/2)(b_x^T D b_x - v_x^T Q^{-1} v_x) + (1/2)(b_y^T D b_y - v_y^T Q^{-1} v_y) + C,$$

where

$$\begin{aligned} D &= \mathbf{diag}(k) \\ Q &= A_2 D A_2^T \\ v_x &= A_2 D b_x \\ v_y &= A_2 D b_y + \bar{c}. \end{aligned}$$

Note that all these terms are affine in k.

We can therefore write down the problem of re-allocating stiffness as

$$\begin{array}{ll} \text{maximize} & b_x^T D b_x - v_x^T Q^{-1} v_x + b_y^T D b_y - v_y^T Q^{-1} v_y \\ \text{subject to} & \mathbf{1}^T k = k^{\text{tot}}, \quad k \succeq 0. \end{array}$$

This is a convex optimization problem. The constraints are obviously convex in k. The objective has two terms which are affine (and therefore concave) in k and two terms which are the negatives of matrix fractionals of affine terms in k (and thus also concave). Thus the objective is concave in k.

Here is another solution to this problem. The stiffness allocation problem is

$$\begin{array}{ll} \text{maximize} & E_{\min}(k) \\ \text{subject to} & \mathbf{1}^T k = k^{\text{tot}}, \quad k \succeq 0. \end{array}$$

Let $(x^\star, y^\star)$ be the equilibrium mass coordinates for stiffness k and let μ be the Lagrange multiplier associated with the equality constraint of k. Then the Lagrangian of the stiffness allocation problem is

$$\begin{array}{rcl} L(k,\mu) & = & (1/2)(x^{\star T} A\, \mathbf{diag}(k) A^T x^\star + y^{\star T} A\, \mathbf{diag}(k) A^T y^\star) + c^T y^\star + \mu(\mathbf{1}^T k - k^{\text{tot}}) \\ & = & \displaystyle\sum_j k_j((1/2)(x^{\star T} a_j a_j^T x^\star + y^{\star T} a_j a_j^T y^\star) + \mu) + c^T y^\star - \mu k^{\text{tot}}, \end{array}$$

where a_j is the jth column of A. However, this is the same as the Lagrangian of the following optimization problem

$$\begin{array}{ll} \text{minimize} & c^T y - \mu k^{\text{tot}} \\ \text{subject to} & (1/2) x^T a_j a_j^T x + (1/2) y^T a_j a_j^T y + \mu \leq 0, \quad j = 1, \ldots, N \\ & x_i = x_i^{\text{fixed}}, \quad i = 1, \ldots, p \\ & y_i = y_i^{\text{fixed}}, \quad i = 1, \ldots, p. \end{array}$$

The value of this optimization problem is equal to the optimal $E_{\min}$. The optimal spring stiffness $k^\star$ is equal to the optimal Lagrange multipliers corresponding to the inequality constraints.

Here is something that doesn't work (but was proposed by a large number of people): alternating between minimizing over x and y and maximizing over k. Some people argued that this would converge to a local optimum of the problem, thus a global optimum since the problem is convex. However, this is not true.

(b) The following code solves the stiffness re-allocation problem.

```
tens_struct_data;

c = g*m;
% Optimum stiffness allocation problem
A1 = A(1:p,:); A2 = A(p+1:n,:); cbar = c(p+1:n);
```

```
cvx_begin
    variable k(N)
    D = diag(k);
    bx = A1'*x_fixed;
    by = A1'*y_fixed;
    vx = A2*D*bx;
    vy = A2*D*by+cbar;
    maximize(bx'*D*bx-matrix_frac(vx,A2*D*A2')+...
                by'*D*by-matrix_frac(vy,A2*D*A2'))
    subject to
        k >= 0; sum(k) == k_tot;
cvx_end

% Compute Emin
Eunif = 0.5*x_unif'*A*diag(k_unif)*A'*x_unif;
Eunif = Eunif + 0.5*y_unif'*A*diag(k_unif)*A'*y_unif;
Eunif = Eunif + c'*y_unif
Emin = 0.5*cvx_optval+c(1:p)'*y_fixed

% Form x and y
xmin = -(A2*D*A2')\(A2*D*A1'*x_fixed);
ymin = -(A2*D*A2')\(A2*D*A1'*y_fixed+cbar);

xopt = zeros(n,1);
xopt(1:p) = x_fixed;
xopt(p+1:n) = xmin;

yopt = zeros(n,1);
yopt(1:p) = y_fixed;
yopt(p+1:n) = ymin;

% Plot optimized structure and structure with uniform stiffness
figure
ind_ex = find(k_unif < 1e-2); %do not show springs with k < 1e-2
Aadj = A(:,setdiff(1:N,ind_ex));
Aadj2 = double(Aadj*Aadj'-diag(diag(Aadj*Aadj')) ~= 0);
gplot(Aadj2,[x_unif y_unif],'o-');
hold on
plot(x_fixed,y_fixed,'ro');
xlabel('x'); ylabel('y');
axis([0.1 0.8 -1.5 1])
```

```
figure
ind_ex = find(k < 1e-2); %do not show springs with k < 1e-2
Aadj = A(:,setdiff(1:N,ind_ex));
Aadj2 = double(Aadj*Aadj'-diag(diag(Aadj*Aadj')) ~= 0);
gplot(Aadj2,[xopt yopt],'o-');
hold on
plot(x_fixed,y_fixed,'ro');
xlabel('x'); ylabel('y');
axis([0.1 0.8 -1.5 1])
hold off
```

The optimal energy is $E_{\min}(k^\star) = 57.84$. On the other hand the minimum energy is 18.37 when stiffness is uniformly allocated. The mass configurations for both uniform and optimal stiffness allocations are shown in figure 4. Springs with stiffness less than 10^{-2} are not shown.

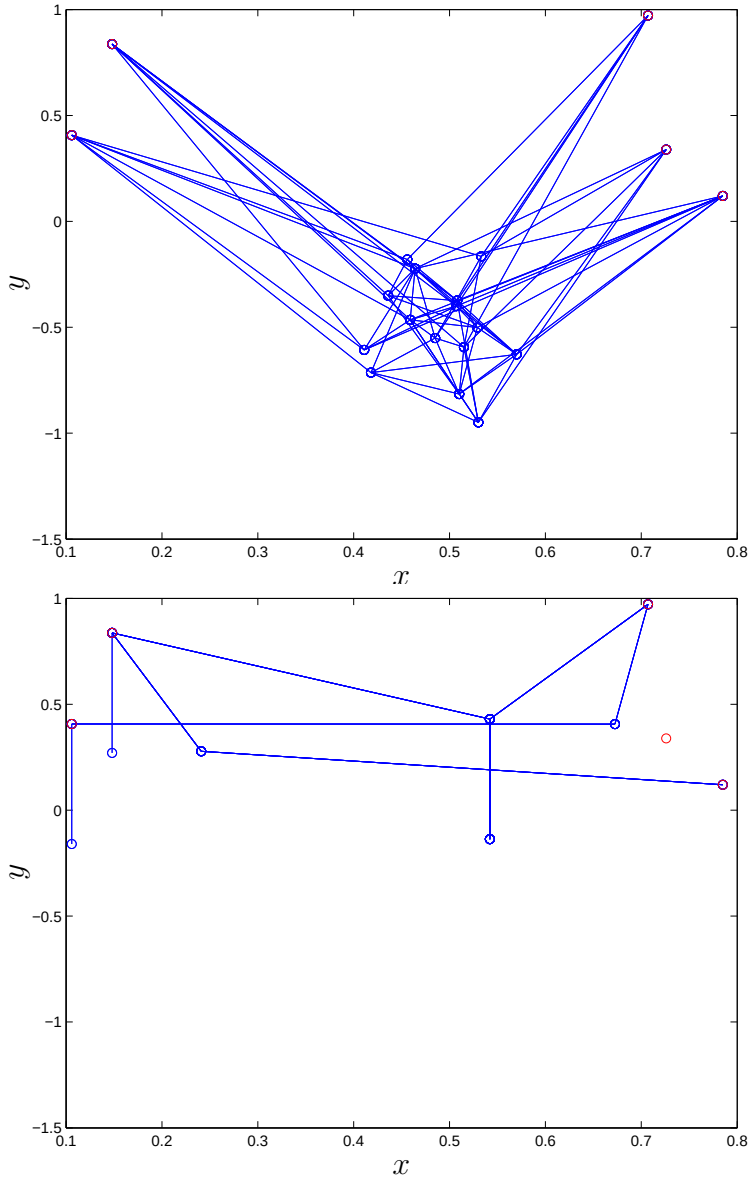


Figure 4: Mass configuration for uniform (top) and optimal (bottom) stiffness allocation.

4. *Identifying a sparse linear dynamical system.* A linear dynamical system has the form

$$x(t+1) = Ax(t) + Bu(t) + w(t), \quad t = 1, \ldots, T-1,$$

where $x(t) \in \mathbf{R}^n$ is the state, $u(t) \in \mathbf{R}^m$ is the input signal, and $w(t) \in \mathbf{R}^n$ is the process noise, at time t. We assume the process noises are IID $\mathcal{N}(0, W)$, where $W \succ 0$ is the covariance matrix. The matrix $A \in \mathbf{R}^{n\times n}$ is called the dynamics matrix or the state transition matrix, and the matrix $B \in \mathbf{R}^{n\times m}$ is called the input matrix.

You are given accurate measurements of the state and input signal, *i.e.*, $x(1), \ldots, x(T)$, $u(1), \ldots, u(T-1)$, and W is known. Your job is to find a state transition matrix $\hat{A}$ and input matrix $\hat{B}$ from these data, that are plausible, and in addition are sparse, *i.e.*, have many zero entries. (The sparser the better.)

By doing this, you are effectively estimating the structure of the dynamical system, *i.e.*, you are determining which components of $x(t)$ and $u(t)$ affect which components of $x(t+1)$. In some applications, this structure might be more interesting than the actual values of the (nonzero) coefficients in $\hat{A}$ and $\hat{B}$.

By plausible, we mean that

$$\sum_{t=1}^{T-1} \left\| W^{-1/2} \left(x(t+1) - \hat{A}x(t) - \hat{B}u(t) \right) \right\|_2^2 \in n(T-1) \pm 2\sqrt{2n(T-1)},$$

where $a \pm b$ means the interval $[a-b, a+b]$. (You can just take this as our definition of plausible. But to explain this choice, we note that when $\hat{A} = A$ and $\hat{B} = B$, the left-hand side is χ^2, with $n(T-1)$ degrees of freedom, and so has mean $n(T-1)$ and standard deviation $\sqrt{2n(T-1)}$.)

(a) Describe a method for finding $\hat{A}$ and $\hat{B}$, based on convex optimization.

We are looking for a *very simple* method, that involves solving *one* convex optimization problem. (There are many extensions of this basic method, that would improve the simple method, *i.e.*, yield sparser $\hat{A}$ and $\hat{B}$ that are still plausible. We're not asking you to describe or implement any of these.)

(b) Carry out your method on the data found in `sparse_lds_data.m`. Give the values of $\hat{A}$ and $\hat{B}$ that you find, and verify that they are plausible.

In the data file, we give you the true values of A and B, so you can evaluate the performance of your method. (Needless to say, you are not allowed to use these values when forming $\hat{A}$ and $\hat{B}$.) Using these true values, give the number of false positives and false negatives in both $\hat{A}$ and $\hat{B}$. A false positive in $\hat{A}$, for example, is an entry that is nonzero, while the corresponding entry in A is zero. A false negative is an entry of $\hat{A}$ that is zero, while the corresponding entry of A is nonzero. To judge whether an entry of $\hat{A}$ (or $\hat{B}$) is nonzero, you can use the test $|\hat{A}_{ij}| \geq 0.01$ (or $|\hat{B}_{ij}| \geq 0.01$).

Solution. The problem can be expressed as

$$\begin{array}{ll} \text{minimize} & \mathbf{card}(\hat{A}) + \mathbf{card}(\hat{B}) \\ \text{subject to} & \sum_{t=1}^{T-1} \left\| W^{-1/2} \left(x(t+1) - \hat{A}x(t) - \hat{B}u(t) \right) \right\|_2^2 \in n(T-1) \pm 2\sqrt{2n(T-1)}, \end{array}$$

where $\mathbf{card}(X)$ is the cardinality (the number of nonzero entries) of matrix X.

However, there are two problems with this: the objective is non-convex, and the lower bound $\sum_{t=1}^{T-1} \left\| W^{-1/2} \left(x(t+1) - \hat{A}x(t) - \hat{B}u(t) \right) \right\|_2^2 \geq n(T-1) - 2\sqrt{2n(T-1)}$ is not a convex constraint. The second problem is dealt with by noting that we can always increase the magnitudes of the implied errors by (for example) multiplying candidate $\hat{A}$ and $\hat{B}$ by a constant. Hence the lower bound can be neglected, without loss of generality.

Unfortunately the **card** function is less comprehensively dealt with. However, we can use the (standard) heuristic of instead minimizing the ℓ_1 norm of the entries of $\hat{A}$ and $\hat{B}$. This gives the convex optimization problem

$$\begin{array}{ll} \text{minimize} & \| \mathbf{vec}(\hat{A}) \|_1 + \| \mathbf{vec}(\hat{B}) \|_1 \\ \text{subject to} & \sum_{t=1}^{T-1} \left\| W^{-1/2} \left(x(t+1) - \hat{A}x(t) - \hat{B}u(t) \right) \right\|_2^2 \leq n(T-1) + 2\sqrt{2n(T-1)}, \end{array}$$

where $\mathbf{vec}(X)$ represents the columns of X concatenated to make a single column vector. Roughly speaking, note that the constraint will always be tight: relaxing the requirement on the implied errors allows more freedom to reduce the ℓ_1 norm of $\hat{A}$ and $\hat{B}$. Thus, in reality, we never need to worry about the lower bound from above as it will be always satisfied.

This problem is easily solved in CVX using the following code

```
% Load problem data.
sparse_lds_data;

fit_tol = sqrt(n*(T-1) + 2*sqrt(2*n*(T-1))); % fit tolerance.
cvx_begin
    variables Ahat(n,n) Bhat(n,m);
    minimize(sum(norms(Ahat, 1)) + sum(norms(Bhat, 1)))
    norm(inv(Whalf)*(xs(:,2:T) - Ahat*xs(:,1:T-1) ...
                     - Bhat*us), 'fro') <= fit_tol;
cvx_end
disp(cvx_status)

% Check lower bound.
fit_tol_m = sqrt(n*(T-1) - 2*sqrt(2*n*(T-1)));
disp(norm(inv(Whalf)*(xs(:,2:T) ...
          - Ahat*xs(:,1:T-1) - Bhat*us), 'fro') >= fit_tol_m);
```

```
% Round near-zero elements to zero.
Ahat = Ahat .* (abs(Ahat) >= 0.01)
Bhat = Bhat .* (abs(Bhat) >= 0.01)

% Display results.
disp(['false positives, Ahat: ' num2str(nnz((Ahat ~= 0) & (A == 0)))])
disp(['false negatives, Ahat: ' num2str(nnz((Ahat == 0) & (A ~= 0)))])
disp(['false positives, Bhat: ' num2str(nnz((Bhat ~= 0) & (B == 0)))])
disp(['false negatives, Bhat: ' num2str(nnz((Bhat == 0) & (B ~= 0)))])
```

With the given problem data, we get 1 false positive and 2 false negatives for $\hat{A}$, and no false positives and 1 false negative for $\hat{B}$. The matrix estimates are

$$\hat{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.1483 & -0.0899 & 0.1375 & -0.0108 & 0 & 0 \\ 0 & 0 & 0 & 0.9329 & 0 & 0 & 0.2868 & 0 \\ 0 & 0.2055 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.0190 & 0.9461 & 0 & 0.8697 \\ 0 & 0 & 0 & 0 & 0.2066 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

and

$$\hat{B} = \begin{bmatrix} -1.4717 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.2832 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1.6363 & -0.0456 & 0 & 0 \\ 0 & 1.4117 & 0 & 0 \\ -0.0936 & 0 & 0 & -0.7755 \\ 0 & -0.5705 & 0 & 0 \end{bmatrix}.$$

Finally, there are lots of methods that will do better than this, usually by taking this as a starting point and 'polishing' the result after that. Several of these have been shown to give fairly reliable, if modest, improvements. You were not required to implement any of these methods.

5. *Minimum energy processor speed scheduling.* A single processor can adjust its speed in each of T time periods, labeled $1, \ldots, T$. Its speed in period t will be denoted s_t, $t = 1, \ldots, T$. The speeds must lie between given (positive) minimum and maximum values, $S^{\min}$ and $S^{\max}$, respectively, and must satisfy a slew-rate limit, $|s_{t+1} - s_t| \leq R$, $t = 1, \ldots, T-1$. (That is, R is the maximum allowed period-to-period change in speed.) The energy consumed by the processor in period t is given by $\phi(s_t)$, where $\phi : \mathbf{R} \to \mathbf{R}$ is increasing and convex. The total energy consumed over all the periods is $E = \sum_{t=1}^{T} \phi(s_t)$.

   The processor must handle n jobs, labeled $1, \ldots, n$. Each job has an availability time $A_i \in \{1, \ldots, T\}$, and a deadline $D_i \in \{1, \ldots, T\}$, with $D_i \geq A_i$. The processor cannot start work on job i until period $t = A_i$, and must complete the job by the end of period D_i. Job i involves a (nonnegative) total work W_i. You can assume that in each time period, there is at least one job available, *i.e.*, for each t, there is at least one i with $A_i \leq t$ and $D_i \geq t$.

   In period t, the processor allocates its effort across the n jobs as θ_t, where $\mathbf{1}^T\theta_t = 1$, $\theta_t \succeq 0$. Here θ_{ti} (the ith component of θ_t) gives the fraction of the processor effort devoted to job i in period t. Respecting the availability and deadline constraints requires that $\theta_{ti} = 0$ for $t < A_i$ or $t > D_i$. To complete the jobs we must have

$$\sum_{t=A_i}^{D_i} \theta_{ti} s_t \geq W_i, \quad i = 1, \ldots, n.$$

   (a) Formulate the problem of choosing the speeds $s_1, \ldots, s_T$, and the allocations $\theta_1, \ldots, \theta_T$, in order to minimize the total energy E, as a convex optimization problem. The problem data are $S^{\min}$, $S^{\max}$, R, ϕ, and the job data, A_i, D_i, W_i, $i = 1, \ldots, n$. Be sure to justify any change of variables, or introduction of new variables, that you use in your formulation.

   (b) Carry out your method on the problem instance described in `proc_sched_data.m`, with quadratic energy function $\phi(s_t) = \alpha + \beta s_t + \gamma s_t^2$. (The parameters α, β, and γ are given in the data file.) Executing this file will also give a plot showing the availability times and deadlines for the jobs.

   Give the energy obtained by your speed profile and allocations. Plot these using the command `bar((s*ones(1,n)).*theta,1,'stacked')`, where s is the $T \times 1$ vector of speeds, and θ is the $T \times n$ matrix of allocations with components θ_{ti}. This will show, at each time period, how much effective speed is allocated to each job. The top of the plot will show the speed s_t. (You don't need to turn in a color version of this plot; B&W is fine.)

Solution. The trick is to work with the variables $x_{ti} = \theta_{ti} s_t$, which must be nonnegative. Let $X \in \mathbf{R}^{T \times n}$ denote the matrix with components x_{ti}. The job completion constraint can be expressed as $X^T\mathbf{1} \succeq W$. The availability and deadline constraints can be expressed as $x_{ti} = 0$ for $t < A_i$ or $t > D_i$, which are linear constraints. The

speed can be expressed as $s = X\mathbf{1}$, a linear function of our variable X. The objective E is clearly a convex function of s (and so, also of X).

Our convex optimization problem is

$$\begin{array}{ll} \text{minimize} & E = \sum_{t=1}^{T} \phi(s_t) \\ \text{subject to} & S^{\min} \preceq s \preceq S^{\max}, \quad s = X\mathbf{1}, \quad X^T\mathbf{1} \succeq W, \quad X \succeq 0 \\ & |s_{t+1} - s_t| \leq R, \quad t = 1, \ldots, T-1 \\ & X_{ti} = 0, \quad t = 1, \ldots, A_i - 1, \quad i = 1, \ldots, n \\ & X_{ti} = 0, \quad t = D_i + 1, \ldots, T, \quad i = 1, \ldots, n \end{array}$$

with variables $s \in \mathbf{R}^T$ and $X \in \mathbf{R}^{T \times n}$. All generalized inequalities here, including $X \succeq 0$, are elementwise nonnegativity. Evidently this is a convex problem.

Once we find an optimal $X^\star$ and $s^\star$, we can recover $\theta_t^\star$ as

$$\theta_{ti}^\star = (1/s_t^\star) x_{ti}^\star, \quad t = 1, \ldots, T, \quad i = 1, \ldots, n.$$

For our data, the availability and deadlines of jobs are plotted in figure 5. The job duration lines show a start time at the beginning of the available time period and a termination time at the very end of the deadline time period. Thus, the length of the line shows the actual duration within which the job is allowed to be completed.

The code that solves the problem is as follows.

```
clear all;
close all;

proc_sched_data;

cvx_begin
    variable X(T,n)
    X >= 0;
    s = sum(X')';
    minimize(sum(alpha+beta*s+gamma*square(s)))
    s >= Smin;
    s <= Smax;
    abs(s(2:end)-s(1:end-1))<=R; % slew rate constraint

    % start/stop constraints
    for i=1:n
        for t=1:A(i)-1
            X(t,i)==0;
        end
        for t=D(i)+1:T
```

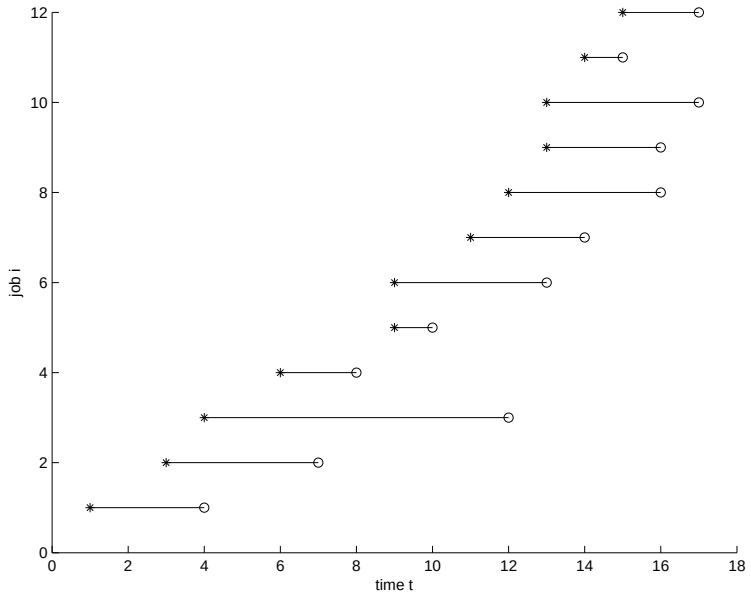


Figure 5: Job availability diagram. Stars indicate the time in which each job becomes available. Open circles indicate the required end of the job, which is at the end of the deadline time period.

```
            X(t,i)==0;
        end
    end

    sum(X)>=W';

cvx_end
theta = X./(s*ones(1,n));

figure;
bar((s*ones(1,n)).*theta,1,'stacked');
xlabel('t');
ylabel('s_t');
```

The optimal total energy is $E = 162.125$, which is obtained by allocating speeds according to the plot shown in figure 6.

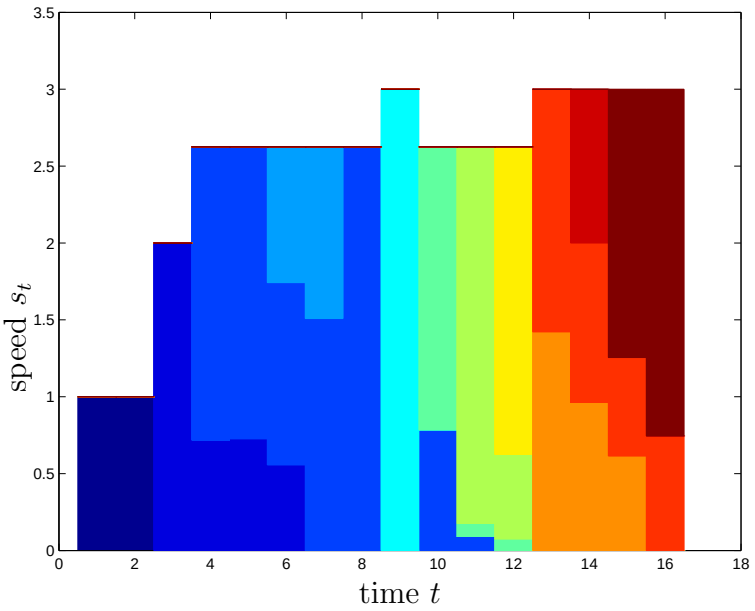


Figure 6: Optimal speed allocation profile. The top of the plot gives the processor speed. The colored regions indicate the portion of the speed allocated to a particular job at each time.

6. *Planning production with uncertain demand.* You must order (nonnegative) amounts $r_1, \ldots, r_m$ of raw materials, which are needed to manufacture (nonnegative) quantities $q_1, \ldots, q_n$ of n different products. To manufacture one unit of product j requires at least A_{ij} units of raw material i, so we must have $r \succeq Aq$. (We will assume that A_{ij} are nonnegative.) The per-unit cost of the raw materials is given by $c \in \mathbf{R}^m_+$, so the total raw material cost is $c^T r$.

   The (nonnegative) demand for product j is denoted d_j; the number of units of product j sold is $s_j = \min\{q_j, d_j\}$. (When $q_j > d_j$, $q_j - d_j$ is the amount of product j produced, but not sold; when $d_j > q_j$, $d_j - q_j$ is the amount of unmet demand.) The revenue from selling the products is $p^T s$, where $p \in \mathbf{R}^n_+$ is the vector of product prices. The profit is $p^T s - c^T r$. (Both d and q are real vectors; their entries need not be integers.)

   You are given A, c, and p. The product demand, however, is not known. Instead, a set of K possible demand vectors, $d^{(1)}, \ldots, d^{(K)}$, with associated probabilities $\pi_1, \ldots, \pi_K$, is given. (These satisfy $\mathbf{1}^T \pi = 1$, $\pi \succeq 0$.)

   You will explore two different optimization problems that arise in choosing r and q (the variables).

   I. Choose r and q ahead of time. You must choose r and q, knowing only the data listed above. (In other words, you must order the raw materials, and commit to producing the chosen quantities of products, before you know the product demand.) The objective is to maximize the expected profit.

   II. Choose r ahead of time, and q after d is known. You must choose r, knowing only the data listed above. Some time after you have chosen r, the demand will become known to you. This means that you will find out which of the K demand vectors is the true demand. Once you know this, you must choose the quantities to be manufactured. (In other words, you must order the raw materials before the product demand is known; but you can choose the mix of products to manufacture after you have learned the true product demand.) The objective is to maximize the expected profit.

   (a) Explain how to formulate each of these problems as a convex optimization problem. Clearly state what the variables are in the problem, what the constraints are, and describe the roles of any auxiliary variables or constraints you introduce.

   (b) Carry out the methods from part (a) on the problem instance with numerical data given in `planning_data.m`. This file will define `A`, `D`, `K`, `c`, `m`, `n`, `p` and `pi`. The K columns of D are the possible demand vectors. For both of the problems described above, give the optimal value of r, and the expected profit.

   Solution. We first consider the case when r and q must be decided before the demand is known. The variables are $r \in \mathbf{R}^m$ and $q \in \mathbf{R}^n$. The cost of the resources is

deterministic; it's just $c^T r$. The revenue from product sales is a random variable, with values

$$p^T \min\{q, d^{(k)}\}, \quad k = 1, \ldots, K,$$

(where $\min\{q, d^{(k)}\}$ is meant elementwise), with associated probabilities $\pi_1, \ldots, \pi_K$. The expected profit is therefore

$$-c^T r + \sum_{k=1}^{K} \pi_k p^T \min\{q, d^{(k)}\}.$$

Our problem is thus

$$\begin{array}{ll} \text{maximize} & -c^T r + \sum_{k=1}^K \pi_k p^T \min\{q, d^{(k)}\} \\ \text{subject to} & q \succeq 0, \quad r \succeq 0, \quad r \succeq Aq, \end{array}$$

with variables r and q. The objective is concave (and piecewise-linear), and the constraints are all linear inequalities, so this is a convex optimization problem.

Now we turn to the case when we must commit to the resource order, but can choose the product mix after we know the demand. In this case we need to have a separate product quantity vector for each possible demand vector. Thus we have variables $r \in \mathbf{R}^m_+$ and

$$q^{(1)}, \ldots, q^{(K)} \in \mathbf{R}^n_+.$$

Here $q^{(k)}$ is the product mix we produce if $d^{(k)}$ turns out to be the actual product demand. Our problem can be formulated as

$$\begin{array}{ll} \text{maximize} & -c^T r + \sum_{k=1}^K \pi_k p^T \min\{q^{(k)}, d^{(k)}\} \\ \text{subject to} & r \succeq 0 \\ & q^{(k)} \succeq 0, \quad r \succeq Aq^{(k)}, \quad k = 1, \ldots, K. \end{array}$$

Note that the variables r and $q^{(1)}, \ldots, q^{(K)}$ must be decided at the same time, taking all of them into account; we cannot optimize them separately. Nor can we decide on r first, without taking the different $q^{(k)}$'s into account. (However, provided we have solved the problem above to get the right r, we can then optimize over q, once we know which demand vector is the true one. But there is no need for this, since we have already worked out the optimal q for each possible demand vector.)

We can simplify this a bit. In this case, we will not produce any excess product, because this wastes resources, and we know the demand before we decide how much to manufacture. So we will have $q \preceq d$, and we can write the problem as

$$\begin{array}{ll} \text{maximize} & -c^T r + \sum_{k=1}^K \pi_k p^T q^{(k)} \\ \text{subject to} & r \succeq 0 \\ & d^{(k)} \succeq q^{(k)} \succeq 0, \quad r \succeq Aq^{(k)}, \quad k = 1, \ldots, K. \end{array}$$

With the values from `planning_data.m`, the optimal values of r for the two problems, respectively, are

$$\begin{aligned} r_{\mathrm{I}} &= (45.5,\ 65.8,\ 44.6,\ 76.7,\ 72.0,\ 77.8,\ 59.3,\ 61.9,\ 83.2,\ 73.1) \\ r_{\mathrm{II}} &= (65.7,\ 59.6,\ 55.2,\ 70.8,\ 69.7,\ 68.4,\ 55.3,\ 57.4,\ 63.8,\ 66.2) \end{aligned}$$

These yield expected profits, respectively, of 114.3 and 133.0.

The `cvx` code that solves this problem is listed below.

```
planning_data;

disp('Problem I.')
cvx_begin
    variable r(m); % amount of each raw material to buy.
    variable q(n); % amount of each product to manufacture.
    r >= 0;
    q >= 0;
    r >= A*q;
    S = min(q*ones(1,K),D);
    maximize(p'*S*pi - c'*r)
cvx_end
rI = r; disp(r); disp(cvx_optval);
profitI = p'*S - c'*r; % save for graphing.

disp('Problem II.')
cvx_begin
    variable r(m); % amount of each raw material to buy.
    variable q(n, K); % product to manufacture, for each demand.
    q >= 0;
    r >= 0;
    r*ones(1,K) >= A*q;
    S = min(q, D);
    maximize(p'*S*pi - c'*r)
cvx_end
rII = r; disp(rII); disp(cvx_optval);
profitII = p'*S - c'*r; % save for graphing.
```

We mention two other optimization problems, that we didn't ask you to explore. First, suppose we committed to buy r_{I} resources under the first plan, but later learned the demand (before production started). The question then is: how suboptimal is the expected profit when using the r_{I} from the first problem instead of r_{II}? For the given numerical instance, our expected profit drops from 133.0 to 128.7.

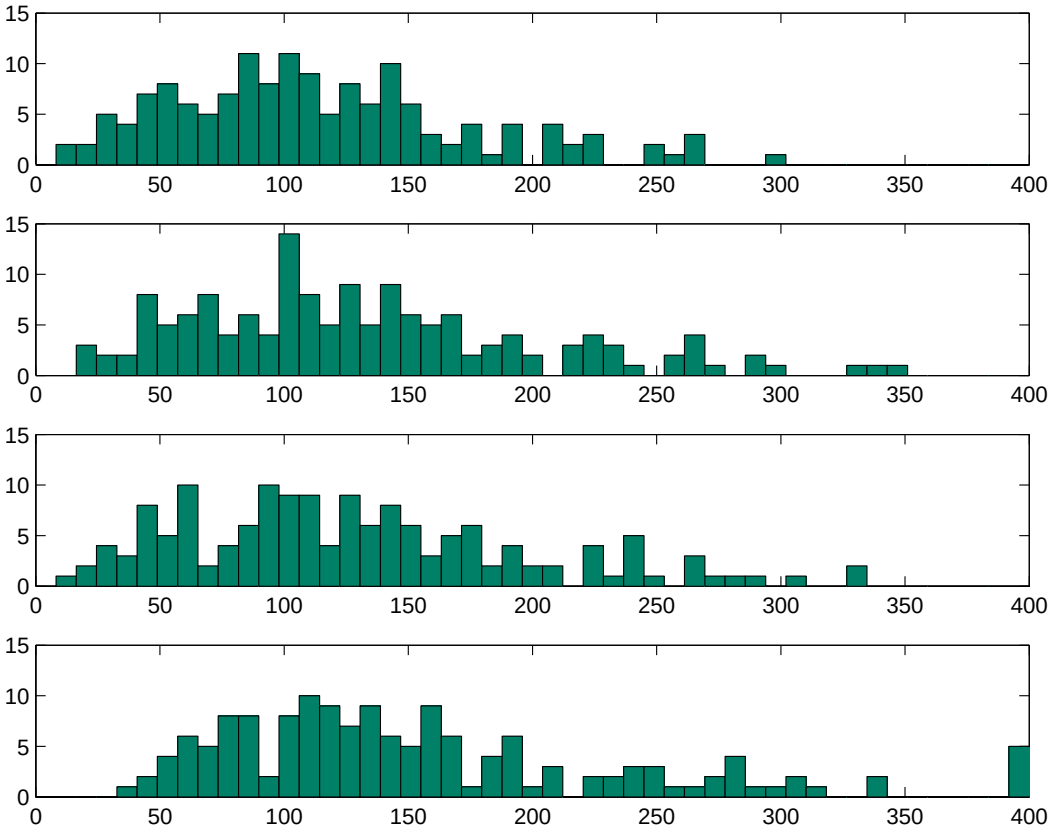
Finally, it's also interesting to look at performance when we know the demand before buying the resources. This is the full prescience, or full information case—we know everything beforehand. In this case we can optimize the choice of r and q for each possible demand vector separately. This obviously gives an upper bound on the profit available when we have less information. In this case, the expected profit with full prescience is 161.4.

The code that solves these two additional problems is listed below.

```
disp('Problem II but with rI.')
r = rI;
cvx_begin
    variable q(n, K); % product to manufacture, for each demand.
    q >= 0;
    r*ones(1,K) >= A*q;
    S = min(q, D);
    maximize(p'*S*pi - c'*r)
cvx_end
disp(p'*S*pi - c'*r);
profitIII = p'*S - c'*r; % save for graphing.

disp('Full prescience.')
cvx_begin
    variable r(m, K); % amount of each raw material to buy, for each demand..
    variable q(n, K); % product to manufacture, for each demand.
    q >= 0;
    r >= 0;
    r >= A*q;
    S = min(q, D);
    maximize((p'*S - c'*r)*pi)
cvx_end
disp((p'*S - c'*r)*pi);
profitIV = p'*S - c'*r; % save for graphing.
```

Histograms showing profits with the different possible outcomes are shown below.


15
10
5
0
0
50
100
150
200
250
300
350
400

7. *Dual of exponential cone.* The exponential cone $K_{\exp} \subseteq \mathbf{R}^3$ is defined as

$$K_{\exp} = \{(x, y, z) \mid y > 0,\ ye^{x/y} \leq z\}.$$

Express the dual cone in the form

$$K_{\exp}^* = \{(u, v, w) \mid \ \ldots \ \textit{your conditions here} \ \ldots\}.$$

Your conditions must be short (certainly no more than one line) and cannot involve any variables other than u, v, and w.

We are not worried here about the fine details of what happens on the boundaries of these cones, so you really needn't worry about it. But we make some comments here for those who do care about such things.

The cone $K_{\exp}$ as defined above is not closed. To obtain its closure, we need to add the points

$$\{(x, y, z) \mid x \leq 0,\ y = 0,\ z \geq 0\}.$$

(This makes no difference, since the dual of a cone is equal to the dual of its closure.)

Solution. The dual cone can be expressed as

$$\begin{aligned} K_{\exp}^* &= \mathbf{cl}\{(u, v, w) \mid u < 0,\ -ue^{v/u} \leq ew\} \\ &= \{(u, v, w) \mid u < 0,\ -ue^{v/u} \leq ew\} \cup \{(0, v, w) \mid v \geq 0,\ w \geq 0\}, \end{aligned}$$

where **cl** means closure. We didn't expect people to add the points on the boundary, *i.e.*, to work out the second set in the second line.

The dual cone can be expressed several other ways as well. The conditions $u < 0$, $-ue^{v/u} \leq ew$ can be expressed as

$$(u/w)\log(-u/w) + v/w - u/w \geq 0,$$

or

$$u\log(-u/w) + v - u \geq 0,$$

or

$$\log(-u/w) \leq 1 - (v/u).$$

Now let's derive the result. For (u, v, w) to be in $K_{\exp}^*$, we need to have $ux+vy+wz \geq 0$ whenever $y > 0$ and $ye^{x/y} \leq z$. Thus, we must have $w \geq 0$; otherwise we can make $ux + vy + wz$ negative by choosing a large enough z. Now let's see what happens when $w = 0$. In this case we need $ux + vy \geq 0$ for all $y > 0$. This happens only if $u = 0$ and $v \geq 0$. So points with

$$u = 0, \quad v \geq 0, \quad w = 0$$

are in $K_{\exp}^*$.

Let's now consider the case $w > 0$. We'll minimize $ux + vy + wz$ over all x, y and z that satisfy $y > 0$ and $ye^{x/y} \leq z$. Since $w > 0$, we minimize over z by taking $z = ye^{x/y}$, which yields

$$ux + vy + wye^{x/y}.$$

Now we will minimize over x. If $u > 0$, this is unbounded below as $x \to -\infty$. If $u = 0$, the minimum is not achieved, but occurs as $x \to -\infty$; the minimum value is then vy. This has a nonnegative minimum value over all $y > 0$ only when $v \geq 0$. Thus we find that points satisfying

$$u = 0, \qquad v \geq 0, \qquad w > 0$$

are in K^*_{exp}.

If $u < 0$, the minimum of $ux + vy + wye^{x/y}$ occurs when its derivative with respect to x vanishes. This leads to $u + we^{x/y} = 0$, *i.e.*, $x = y \log(-u/w)$. For this value of x the expression above becomes

$$y(u \log(-u/w) + v - u).$$

Now we minimize over $y > 0$. We get $-\infty$ if $u \log(-u/w) + v - u < 0$; we get 0 if $u \log(-u/w) + v - u \geq 0$.

So finally, we have our condition:

$$u \log(-u/w) + v - u \geq 0, \quad u < 0, \quad w > 0.$$

Dividing by u and taking the exponential, we can write this as

$$-ue^{v/u} \leq ew, \qquad u < 0.$$

(The condition $w > 0$ is implied by these two conditions.)

Finally, then, we have

$$K^*_{\text{exp}} = \{(u, v, w) \mid u < 0,\ -ue^{v/u} \leq ew\} \cup \{(0, v, w) \mid v \geq 0,\ w \geq 0\}.$$