EE364a: Convex Optimization I M. Udell, R. Takapoui
August 16–17, 2013

Final Exam Solutions

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

You may use any books, notes, or computer programs (*e.g.*, Matlab, CVX), but you may not discuss the exam with anyone until August 17, 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 or 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 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/data_for_final/filename.m`

with your browser.

All problems have equal weight.

Be sure you are using the most recent version of CVX, which is Version 2.0 (beta), build 1002. You can check this using the command `cvx_version`.

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

1. *Signal reconstruction from level information.* Suppose a signal $x \in \mathbf{R}^n$ lies in a subspace $S \subseteq \mathbf{R}^n$. We cannot observe the signal directly, but a level detector reveals the times at which a the signal is high, low, or small. Our goal is to use this information to estimate x.

   Let the detector function $d : \mathbf{R} \to \{-1, 0, 1\}$ be defined by

$$d(a) = \begin{cases} +1 & a \geq 1 \\ 0 & |a| < 1 \\ -1 & a \leq -1. \end{cases}$$

   We receive only $l \in \mathbf{R}^n$, where $l_i = d(x_i)$. We refer to l as *level data*. We say that a signal $z \in \mathbf{R}^n$ is consistent with the level data l if $d(z_i) = l_i$, $i = 1, \ldots, n$.

   Throughout, let $S = \mathcal{R}(D) = \{Dw \mid w \in \mathbf{R}^p\}$, where $D \in \mathbf{R}^{n \times p}$ is a given matrix. That is, the columns of D form a basis for S.

   (a) Show how to find the estimate $\hat{x} \in S$ of minimum norm $\|\hat{x}\|_2$ that is consistent with the level data l using convex optimization. (You may assume that the signal never achieves values of exactly 1 or -1.) Your answer should be in terms of the problem data D and l.

   (b) Implement your approach from part (a) using the data contained in `subspace_level_data.m`. For comparison, the data file also contains the original signal `x_orig`. (Needless to say, you may not use it in forming your estimate.) Plot your estimate, along with the original signal, in the same figure.

   Solution.

   (a) We define the index sets

$$\begin{aligned} \mathcal{H}(l) &= \{i \mid l_i = 1\} \\ \mathcal{L}(l) &= \{i \mid l_i = -1\} \\ \mathcal{S}(l) &= \{i \mid l_i = 0\} \end{aligned}$$

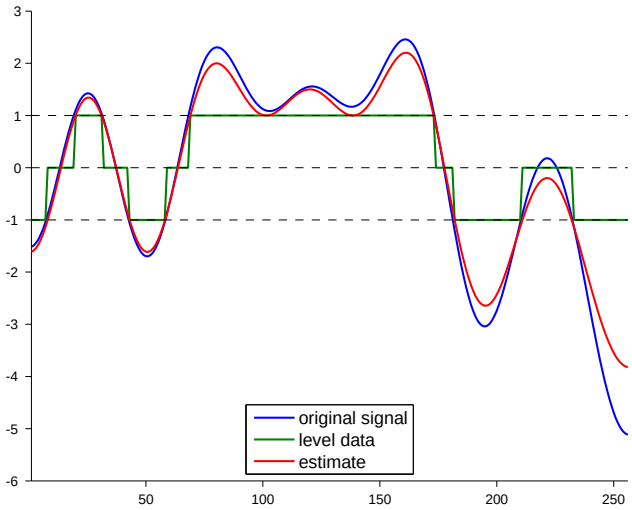
   These contain the times at which the signal is high, low, and small, respectively. The convex problem to be solved is

$$\begin{array}{ll} \text{minimize} & \|x\|_2 \\ \text{subject to} & x = Dw \\ & x_i \geq 1, \quad i \in \mathcal{H}(l) \\ & x_i \leq -1, \quad i \in \mathcal{L}(l) \\ & |x_i| \leq 1, \quad i \in \mathcal{S}(l) \end{array}$$

   in variables $x \in \mathbf{R}^n$ and $w \in \mathbf{R}^p$. Note that since we assume $|x_i| \neq 1$, the strict inequality has been relaxed for those indices in $\mathcal{S}$.

(b) The following code implements the reconstruction method of part (a).

```
% Signal reconstruction level data

subspace_level_data

high = l == 1;
low = l == -1;
small = l == 0;

cvx_begin
variables x(n) w(p)
    minimize( norm(x) )
    subject to
        x == D*w
        x(high) >= 1;
        x(low) <= -1;
        abs(x(small)) <= 1;
cvx_end

figure(1); clf
plot(1:n,x_orig,1:n,l,1:n,x,'linewidth',1.5); hold on
plot([1 n],[1 1],'k--');
plot([1 n],[0 0],'k--');
plot([1 n],[-1 -1],'k--');  hold off
leg = legend('original signal','level data','estimate');
set(leg,'location','south','fontsize',13)
set(gca,'box','off','tickdir','out')
xlim([1 n])
```


3
2
1
0
-1
-2
-3
-4
-5
-6
50
100
150
200
250
original signal
level data
estimate

2. *Color rendering in an imaging system.* In this problem we want to reproduce the colors in a visual scene with the use of a given camera and projector. We do this by controlling how output from the camera is transformed into input to the projector.

   The visual scene is divided into a collection of pixels. Our imaging system behaves the same at each pixel, so we focus on reconstructing a single pixel. Every pixel in the scene is characterized by its intensity across the visible spectrum of light. For the spectral resolution of the human eye, it's sufficient to consider L distinct frequencies at which we measure the intensity. Thus, the scene at each pixel is represented by a signal $x \in \mathbf{R}^L$, where x_i is the intensity of light at the ith frequency.

   The camera records at each pixel through a set of M photodiodes. The ith photodiode is characterized by $d_i \in \mathbf{R}^L$, which governs its linear response to the scene. Hence, at each pixel, we can write the camera's response to a signal x as $D^T x$, where $D = [d_1 \ \cdots \ d_M]$.

   The projector emits light at each pixel through a set of N LEDs, with emission spectra $g_j \in \mathbf{R}^L$. Given input $y \in \mathbf{R}^N$, the combined output is linear and given by Gy, where $G = [g_1 \ \cdots \ g_N]$.

   An *imaging system* is defined by a matrix $P \in \mathbf{R}^{N \times M}$ mapping the camera output $D^T x$ to the projector input $y = PD^T x$. The resulting image is $GPD^T x$. The LEDs can only have nonnegative activation, so we require $P_{ij} \geq 0$.

   For a signal x, we measure the reconstruction error as $\|E^T(x - GPD^T x)\|_2$, where $E \in \mathbf{R}^{3 \times L}$ contains the frequency responses of the three types of color receptors in the human eye. Here we will be concerned with the *worst-case reconstruction error*, given by the largest reconstruction error obtained over all signals satisfying $\|x\|_2 \leq 1$.

   (a) Write the problem of finding the imaging system P that minimizes the worst-case reconstruction error as a convex optimization problem. Your answer should be in terms of problem data G, D, and E.

   (b) Use the data provided in `img_sys_data.m` to find the imaging system matrix P that minimizes the worst case reconstruction error. Report the optimal value.

   (c) To reduce manufacturing costs, we would like to use a small number of LEDs and photodiodes. If all the entries of P in column i are zero, then we may leave out photodiode i from the camera; similarly, if all the entries of P in row j are zero, then we may leave out LED j from the projector. To find a cheaper system that still produces a faithful image, minimize $\|P^T \mathbf{1}\|_1 + \|P\mathbf{1}\|_1$ subject to the constraint that the worst-case reconstruction error does not exceed 1. Plot the resulting matrix P using the command `spy(abs(P)>0.01)`. With this imaging system, how many photodiodes and how many LEDs will be required?

   Solution.

(a) Recall that, by definition of the matrix norm,

$$\sup_{\|x\|_2 \leq 1} \|E^T(I - GPD^T)x\|_2 = \|E^T(I - GPD^T)\|_2.$$

This is a convex function of P. The optimization problem is thus

$$\begin{array}{ll} \text{minimize} & \|E^T(I - GPD^T)\|_2 \\ \text{subject to} & P_{ij} \geq 0. \end{array}$$

(b) The following code solves the problem:

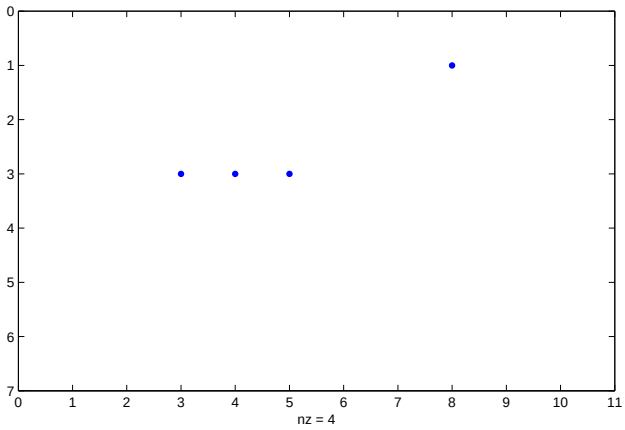
```
% Designing an optimal imaging system (b)
img_sys_data

cvx_begin
    variable P(N,M)
    minimize(norm(E'*(eye(L)-G*P*D'),2))
    subject to
    P >= 0
cvx_end

% optimal value:
norm(E'*(eye(L)-G*P*D'),2)
```

The optimal value is 0.935.

(c) The following code solves the problem:

```
% Designing an optimal imaging system (c)
img_sys_data

cvx_begin
    variable P(N,M)
    minimize( norm(sum(P),1)+norm(sum(P'),1) )
    subject to
    norm(E'*(eye(L)-G*P*D'),2) <= 1
    P >= 0
cvx_end

% visualize result
spy(abs(P)>0.01)
```

The sparsity of the matrix (shown below) indicates that 4 photodiodes and 2 LEDs are required.


0
1
2
3
4
5
6
7
0
1
2
3
4
5
6
7
8
9
10
11
nz = 4

3. *Imaging under Poisson noise.* In some imaging systems, data acquisition is modeled with Poisson noise. Let $x \in \mathbf{R}^n_{++}$ be an underlying signal. (The values in x could represent the tissue density sampled along some direction in space, for example.) In acquiring our image, we take m independent measurements

$$y_i \sim \text{Pois}(a_i^T x), \qquad i = 1, \ldots, m,$$

where $a_i \in \mathbf{R}^n_{++}$ is a known, positive vector that depends on the device. In words, we measure m independent Poisson random variables, each with mean given by some weighted sum of the underlying signal. Recall that if $z \sim \text{Pois}(\mu)$ it has probability mass function

$$\mathbf{Prob}(z = k) = \frac{e^{-\mu}\mu^k}{k!}, \qquad k = 0, 1, 2, 3, \ldots$$

For brevity, we let the matrix A have rows a_i^T.

(a) Show how to compute the maximum likelihood estimate of the signal x, given observations y and the measurement matrix A, using convex optimization.

(b) Suppose that $m = n$ and the measurement matrix $A = I$. Analytically determine the solution to the maximum likelihood problem in terms of the observations y_i.

(c) Suppose we know *a priori* that the signal contains a few regions of roughly constant intensity. To enforce this property in the solution, we augment the negative log-likelihood with an ℓ_1-norm penalty on the first difference of the signal. The criterion to minimize becomes

$$-l(x) + \lambda J(x),$$

where $l(x)$ is the log-likelihood function from part (a), $\lambda \geq 0$ is a regularization parameter, and

$$J(x) = \sum_{i=1}^{n-1} |x_{i+1} - x_i|.$$

What happens to the solution $x^\star$ as $\lambda \to \infty$? Your answer should be in terms of y and A.

(d) Implement the procedure from part (c) to recover the signal x using the data given in `pois_reg_data.m`. Take $\lambda \in \{0.2, 0.6, 1.5\}$ and plot your result for each value.

Solution.

(a) The log-likelihood of the sample is

$$l(x) = \sum_{i=1}^{m} \left[y_i \log(a_i^T x) - a_i^T x \right].$$

Given the data y and measurement matrix A, this function is concave in the parameters x. To reconstruct the signal, we simply maximize $l(x)$.

(b) In this case, we have

$$l(x) = \sum_{i=1}^{m} \left[y_i \log(x_i) - x_i \right],$$

which is now separable. The first order optimality condition gives

$$0 = \frac{\partial l(x)}{\partial x_i} = y_i/x_i - 1 \implies x_i = y_i.$$

The maximum likelihood estimates are simply the observed quantities.

(c) Note that $J(x) \geq 0$, with equality if and only if x is a constant signal. Since $l(x) \geq 0$ as well, the objective will become arbitrarily large with λ unless x is constant. In this case, we have $x = t\mathbf{1}$ for some $t \in \mathbf{R}_+$ and the objective becomes

$$\begin{aligned} l(t\mathbf{1}) &= \sum_{i=1}^{m} \left[y_i \log(t \cdot a_i^T \mathbf{1}) - t \cdot a_i^T \mathbf{1} \right] \\ &= \sum_{i=1}^{m} \left[y_i \log t + \log a_i^T \mathbf{1} - t \cdot a_i^T \mathbf{1} \right] \\ &= \log t \sum_{i=1}^{m} y_i - t \sum_{i=1}^{m} a_i^T \mathbf{1} + \sum_{i=1}^{m} \log a_i^T \mathbf{1}. \end{aligned}$$

Since $l(x)$ is concave, it remains concave when restricted to this line. We maximize over t by setting its derivative to zero:

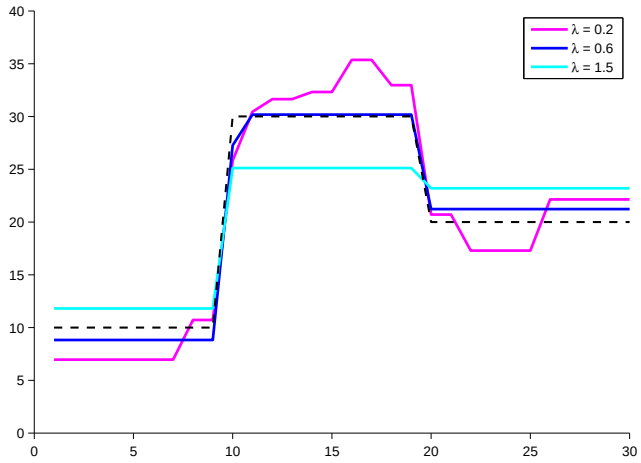
$$0 = \frac{1}{t} \mathbf{1}^T y - \mathbf{1}^T A \mathbf{1} \implies t = \frac{\mathbf{1}^T y}{\mathbf{1}^T A \mathbf{1}} = \frac{\sum_{i=1}^{m} y_i}{\sum_{i,j} A_{ij}}.$$

(d) The following code implements the regularized maximum likelihood procedure.

```
% Imaging under Poisson noise

pois_reg_data;

figure(1); clf; hold on

lambdas = [0.2 0.6 1.5];
colors = {'m','b','c'};
ph = zeros(3,1);
label = cell(3,1);

for i = 1:3
    lambda = lambdas(i);
```

```
    cvx_begin quiet
        variable z(n)
        loglik = y'*log(A*z) - sum(A*z);
        penalty = norm( z(2:end) - z(1:end-1) ,1);
        minimize(-loglik + lambda*penalty)
    cvx_end
    ph(i) = plot(z,colors{i},'linewidth',2);
    label{i} = ['\lambda = ' num2str(lambda)];

end

plot(x,'k--','linewidth',1.5)
legend(ph,label);
set(gca,'box','off','tickdir','out')
ylim([0 40])
```

The resulting estimates are plotted below. The dashed curve depicts the true signal.



4. *Sparse approximate eigenvectors.* Let $A \in \mathbf{S}^n$ be a given matrix. It is well known that any solution $x^\star$ to the problem

$$\begin{array}{ll} \text{maximize} & x^T A x \\ \text{subject to} & \|x\|_2 = 1 \end{array}$$

is an eigenvector of A corresponding to its largest eigenvalue. (If A is a covariance matrix, this problem can be interpreted as finding a normalized linear combination of the variables that has maximal variance, also known as principal component analysis.) In this exercise, we seek a sparse *approximate* eigenvector. By this we mean a unit-length vector x that achieves a large value of $x^T A x$, while simultaneously possessing few nonzero elements.

(a) Consider the following problem in variable $X \in \mathbf{S}^n$:

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(XA) \\ \text{subject to} & \mathbf{Tr}(X) = 1 \\ & \mathbf{Rank}(X) = 1 \\ & X \succeq 0. \end{array}$$

Show that its solution must have the form $X^\star = x^\star (x^\star)^T$, where $x^\star$ is a unit-length eigenvector of A corresponding to its largest eigenvalue. Explain how you would recover $x^\star$ from the optimal $X^\star$. (This establishes the equivalence of the two optimization problems presented above.)

(b) We can impose sparsity on our approximate eigenvector by requiring that $\mathbf{card}(X) \leq k^2$, for some integer k. (With matrix argument, $\mathbf{card}(\cdot)$ returns the total number of nonzero elements.) Since the rank and cardinality constraints are not convex, we relax our problem as follows:

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(XA) \\ \text{subject to} & \mathbf{Tr}(X) = 1 \\ & \mathbf{1}^T |X| \mathbf{1} \leq k \\ & X \succeq 0. \end{array}$$

Here, $|\cdot|$ denotes element-wise absolute value. Show that this is indeed a relaxation of the problem in part (a) when it is supplemented with the cardinality constraint. That is, if

$$\begin{aligned} \mathcal{R}_1 &= \{X \in \mathbf{S}^n \mid X \succeq 0,\ \mathbf{Tr}(X) = 1,\ \mathbf{Rank}(X) = 1,\ \mathbf{card}(X) \leq k^2\} \\ \mathcal{R}_2 &= \{X \in \mathbf{S}^n \mid X \succeq 0,\ \mathbf{Tr}(X) = 1,\ \mathbf{1}^T |X| \mathbf{1} \leq k\}, \end{aligned}$$

show that $\mathcal{R}_1 \subseteq \mathcal{R}_2$.

(c) In this part we will apply the procedure above to the simulated data contained in `sparse_eigs_sdp_data.m`. Let $x_0 \in \mathbf{R}^n$ be a sparse vector. The matrix under consideration is

$$A = \lambda X_0 + Z,$$

where λ is a large positive number, $X_0 = x_0 x_0^T$, and Z is a small, symmetric perturbation. Use the method from part (b) with $k = 7.5$ to estimate x_0. For comparison, the data file also finds the leading eigenvector of A using Matlab's `eigs` command.

We are interested in identifying the support of x_0, *i.e.*, the indices at which x_0 is nonzero. The data file contains code to compute the numbers of *true* and *false positives*. True positives occur at indices at which both $\hat{x}_0$ and x_0 are nonzero. False positives occur at indices at which $\hat{x}_0$ is nonzero, but x_0 is in fact zero. Report the numbers of true and false positives generated by your estimate and by the leading eigenvector of A. (That's four numbers in total.) To judge whether a number a is nonzero, use the criterion $|a| > 0.01$.

Solution.

(a) Suppose X is feasible. Since X is symmetric rank 1, it is unitarily diagonalizable and can be written as $X = a \cdot vv^T$, where $v \in \mathbf{R}^n$ and $a \in \mathbf{R}$ is a nonzero eigenvalue. Now consider the trace constraint.

$$1 = \mathbf{Tr}(X) = a\,\mathbf{Tr}(vv^T) = a\sum_{i=1}^n v_i^2 \implies a = \frac{1}{\|v\|_2^2}$$

It follows that we can write $X = xx^T$, where $x = v/\|v\|_2$ has unit norm.
The objective is then

$$\mathbf{Tr}(XA) = \mathbf{Tr}(xx^TA) = \mathbf{Tr}(x^TAx) = x^TAx,$$

where we have used the fact that the trace is invariant under cyclic permutations. We can recover $x^\star$ by finding the leading eigenvector (or singular vector) of $X^\star$.

(b) Recall that there is a one-to-one correspondence between feasible matrices in (a) and unit length vectors x, where $X = xx^T$. All such matrices satisfy $X \succeq 0$. Furthermore, since $X_{ij} = x_i x_j$, $\mathbf{card}(X) \leq k^2 \iff \mathbf{card}(x) \leq k$. In this case, we have $\|x\|_1 \leq \sqrt{k}\|x\|_2 = \sqrt{k}$. (This can be seen via an application of the Cauchy-Schwarz inequality.)

Now,

$$\mathbf{1}^T|X|\mathbf{1} = \mathbf{1}^T|x||x|^T\mathbf{1} = \|x\|_1^2.$$

Therefore, if X is feasible for the problem in part (b), then $\mathbf{card}(X) \leq k^2 \implies \mathbf{1}^T|X|\mathbf{1} \leq k$.

(c) The following code carries out the procedure outline above.

```
% Sparse approximate eigenvectors
%% Generate Data

n = 20;
randn('state',1)
x0 = sign(randn(n,1));
sp = [1 3 5 6 7 8 11 14 15 16 17 19];
x0(sp) = 0;
x0 = x0/norm(x0);
X0 = x0*x0';

noise = randn(n,n);
Z = (noise + noise')/sqrt(2);

lambda = 12;
A = lambda*X0 + Z;

k = 7.5;

%% Your code goes here
% It should produce an estimate x0_hat

cvx_begin
    variable X(n,n) symmetric
    maximize( trace(X*A) )
    subject to
    trace(X) == 1
    X == semidefinite(n)
    norm(vec(X),1) <= k
cvx_end

% recover x0 from solution
[x0_hat,~] = svds(X,1);

%% Analyze results

%leading eigenvector of A (for comparison)
[v,~] = eigs(A,1,'LA');

plot([x0,-x0_hat,v]) %signs may be flipped

% Compute numbers true and false positives
```

```
tol = 0.01;
TP = sum(  x0 & (abs(x0_hat)>tol) );
FP = sum( ~x0 & (abs(x0_hat)>tol) );

TP2 = sum(  x0 & (abs(v)>tol) );
FP2 = sum( ~x0 & (abs(v)>tol) );
```

Both estimates yield 8 true positives. However, the estimate derived from the semidefinite program yields only 2 false positives, while the principal eigenvector of A yields 11.

5. *Risk budget allocation.* Suppose an amount $x_i > 0$ is invested in n assets, labeled $i = 1, ..., n$, with asset return covariance matrix $\Sigma \in \mathbf{S}^n_{++}$. We define the *risk* of the investments as the standard deviation of the total return, $R(x) = (x^T \Sigma x)^{1/2}$.

   We define the (relative) *risk contribution* of asset i (in the portfolio x) as

$$\rho_i = \frac{\partial \log R(x)}{\partial \log x_i} = \frac{\partial R(x)}{R(x)} \frac{x_i}{\partial x_i}, \quad i = 1, ..., n.$$

   Thus ρ_i gives the fractional increase in risk per fractional increase in investment i. We can express the risk contributions as

$$\rho_i = \frac{x_i (\Sigma x)_i}{x^T \Sigma x}, \quad i = 1, \ldots, n,$$

   from which we see that $\sum_{i=1}^n \rho_i = 1$. For general x, we can have $\rho_i < 0$, which means that a small increase in investment i decreases the risk. Desirable investment choices have $\rho_i > 0$, in which case we can interpret ρ_i as the fraction of the total risk contributed by the investment in asset i. Note that the risk contributions are homogeneous of degree zero, *i.e.*, scaling x by a positive constant does not affect ρ_i.

   In the *risk budget allocation problem*, we are given Σ and a set of desired risk contributions $\rho_i^{\text{des}} > 0$ with $\mathbf{1}^T \rho^{\text{des}} = 1$; the goal is to find an investment mix $x \succ 0$, $\mathbf{1}^T x = 1$, with these risk contributions. When $\rho^{\text{des}} = (1/n)\mathbf{1}$, the problem is to find an investment mix that achieves so-called *risk parity*.

   (a) Explain how to solve the risk budget allocation problem using convex optimization. Is the solution necessarily unique? *Hint.* Minimize $(1/2)x^T \Sigma x - \sum_{i=1}^n \rho_i^{\text{des}} \log x_i$.

   (b) Find the investment mix that achieves risk parity for the return covariance matrix

$$\Sigma = \begin{pmatrix} 6.1 & 2.9 & -0.8 & 0.1 \\ 2.9 & 4.3 & -0.3 & 0.9 \\ -0.8 & -0.3 & 1.2 & -0.7 \\ 0.1 & 0.9 & -0.7 & 2.3 \end{pmatrix}.$$

   For your convenience, this is contained in `risk_alloc_data.m`.

Solution.

(a) Consider the following problem:

$$\text{minimize} \quad (1/2)x^T \Sigma x - \sum_{i=1}^n \rho_i^{\text{des}} \log x_i$$

which is an unconstrained convex problem (with implicit constraint $x_i > 0$). The optimality condition for this problem is:

$$\nabla f(x) = \Sigma x - \text{diag}(x)^{-1} \rho^{\text{des}} = 0.$$

Hence, at the optimal point x^*, we will have:

$$x_i^*(\Sigma x^*)_i = \rho_i^{\text{des}}.$$

The constraint $\mathbf{1}^T\rho^{\text{des}} = 1$ implies that $x^{*T}\Sigma x^* = \sum_{i=1}^n x_i^*(\Sigma x^*)_i = 1$. Therefore, we also have

$$\rho_i^{\text{des}} = \frac{x_i^*(\Sigma x^*)_i}{x^{*T}\Sigma x^*}.$$

Note that this equation is homogeneous in x. So choosing $\hat{x} = x^*/(\mathbf{1}^T x^*)$ will do the trick. Since the objective function is strictly convex, the minimizer is unique; hence, the optimal investment mix is unique.

(b) The following code implements the solution:

```
risk_alloc_data
n = size(Sigma,1);
rho = ones(n,1)/n;

cvx_begin
    variable x(n)
    minimize (1/2*quad_form(x,Sigma) - log( geo_mean(x)))
%     could also use:
%     minimize (1/2*quad_form(x,Sigma) - rho'*log(x))
cvx_end
x_hat = x/sum(x);

% verify risk parity
x_hat.*(Sigma*x_hat)./quad_form(x_hat,Sigma)

% compare with uniform allocation
u = ones(n,1)/n;
u.*(Sigma*u)./quad_form(u,Sigma)
```

The optimal investment mix is: $\hat{x} = (.1377, .1134, .4759, .2731)$.

6. *Optimal module placement.* Efficient utilization of space is an important problem in circuit design. This problem concerns the placement of electronic components (such as transistors, microprocessors, *etc.*) onto a two-dimensional substrate or board.

   Suppose you are given a set of n rectangular modules named $M_1, M_2, ..., M_n$ and a corresponding list of n triplets (a_i, r_i, s_i). The triplet of numbers (a_i, r_i, s_i), with $r_i \leq s_i$, specifies the area and aspect ratio constraints for M_i; if w_i and h_i are the width and height of M_i, respectively, they must satisfy $w_i h_i = a_i$ and $r_i \leq h_i/w_i \leq s_i$. Our goal is to place these modules within a rectangular board of minimum area, such that no two modules overlap.

   In this instance of the problem, with $n = 5$, we insist that the following constraints be satisfied:

   - M_1 and M_2 are to the right of M_3, M_4, and M_5,
   - M_1 is above M_2,
   - M_3 is below M_4 and M_5,
   - M_4 is to the left of M_5.

   Note that these constraints ensure the modules do not overlap. The *module placement problem* is to find relative module locations consistent with the above constraints, such than the enveloping rectangular board has minimum area.

   (a) Formulate the module placement problem as a convex convex optimization problem. *Hint.* You will need to introduce parameterizations for the module locations and sizes. Try using the coordinates of their lower left corners, as well as their heights and widths.

   (b) Solve the optimal module placement problem using the data provided in `module_placement_data.m`. Report the area of the enveloping rectangular board.

Solution.

(a) Using the hint, we can formulate the problem as a geometric program, since the objective and inequality constraints will be posynomial and the equality constraints will be monomial. Let $x, y, h, w \in \mathbf{R}^5$ be vectors such M_i has lower left corner (x_i, y_i), height h_i, and width w_i. Also let H and W be the height and width of the enveloping board, respectively. The optimal module placement problem becomes:

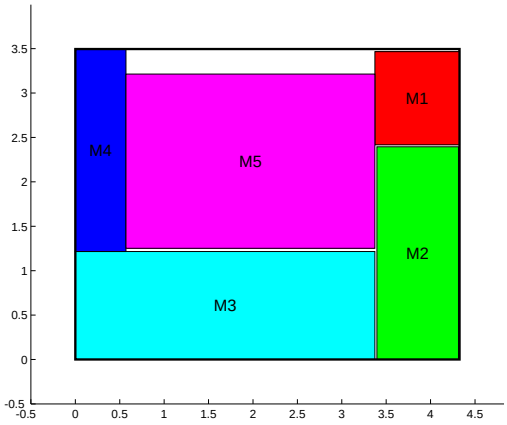
$$\begin{array}{ll} \text{minimize} & HW \\ \text{subject to} & h_i w_i = a_i \quad i = 1, \ldots, n \\ & r_i \leq h_i/w_i \leq s_i \quad i = 1, \ldots, n \\ & x_3 + w_3 \leq x_1 \\ & \textit{etc.} \end{array}$$

(b) The following code solves the geometric program from part (a).

```
% Optimal module placement
module_placement_data

cvx_begin gp
    variables x(n) y(n) w(n) h(n) W  H
    minimize W*H
    subject to
        w.*h == a; %area constraints
        r <= h./w <= s; %aspect ratio constraints
        x/W + w/W <= 1; %components inside the rectangle (horizontal)
        y/H + h/H <= 1; %components inside the rectangle (vertical)
        y(1) >= y(2) + h(2); %M1 is above M2
        x(4)+w(4) <= x(5); %M4 is to the left of M5
        x(3:5) + w(3:5) <= x(1); %M1 is to the right of M3,M4,M5
        x(3:5) + w(3:5) <= x(2); %M2 is to the right of M3,M4,M5
        y(3) + h(3) <= y(4:5);
cvx_end

% report area
W*H

% plot layout
colors = {'r','g','c','b','m'};
figure(1); clf; hold on
axis([-0.5 W+0.5 -0.5 H+0.5]);
for i=1:n
    rec = rectangle('Position', [x(i) y(i) w(i) h(i)]);
    set(rec,'facecolor',colors{i})
    t = text( x(i)+ w(i)/2, y(i) + h(i)/2,[ 'M' num2str(i)]);
    set(t,'fontsize',14,'HorizontalAlignment','center');
end
rectangle('Position', [0 0 W H],'linewidth',2)
hold off
axis image;
```

The area of the enveloping rectangular board is 15.125.

The optimal layout looks like this:


M1
M2
M3
M4
M5
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5

7. *Numerical linear algebra with a fast operator.* Suppose we wish to solve a large linear system of the form

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix},$$

where $A \in \mathbf{R}^{n\times n}$, $B \in \mathbf{R}^{n\times p}$, $C \in \mathbf{R}^{p\times p}$, $x, f \in \mathbf{R}^n$, and $y, g \in \mathbf{R}^p$. Here we assume that $n \gg p \gg \log n$. The entries of A are unknown to us, but the linear operator and its inverse are available as fast subroutines that can be applied to a vector in order $n \log n$ flops. (Recall that usual matrix-vector products require order n^2 flops.)

(a) Explain how to solve this system efficiently, and state the dominant term in the flop count for your procedure. Your answer should be in terms of n and p. (You may assume any square matrices you need are invertible.)

(b) The file `fast_op_data.m` contains code to generate a random instance of the problem for dimensions n and p. In addition, `my_dct.m` and `my_idct.m`, are functions that apply A and A^{-1}, respectively. (These functions compute a vector's *discrete cosine transform*, commonly used in signal and image processing, but it's not necessary to understand what this is.) Fix $p = 100$. For each $n \in \{1000, 2500, 5000, 7500, 10000\}$ generate (at least) 10 random problem instances, and solve them using your method from part (a). Plot the average execution time versus n. Does it look as you would expect?

Hints.

- The functions `my_dct` and `my_idct` accept both vector and matrix arguments.
- To time a section of code in Matlab, use the following commands:

```
t0 = tic;
⋮
t  = toc(t0);
```

The variable `t` then contains the execution time in seconds.

Solution.

(a) We can solve this system via block elimination. We first compute

$$y = (C - B^T A^{-1} B)^{-1}(g - B^T A^{-1} f).$$

Note that $A^{-1}B$ can be formed by concatenating the results of applying A^{-1} to each of the columns of B. Next we use the result to find

$$x = A^{-1}(f - By).$$

Because A^{-1} can be applied in order $n \log n$ flops, the dominant computational cost is incurred during the multiplication of B^T and $A^{-1}B$, with complexity $2np^2$.

(b) The following code carries out the requested numerical experiment.

```
% Numerical linear algebra with a fast operator

p = 100;
N = [1000 2500 5000 7500 10000];
K = length(N);
exec_time = zeros(K,1);
nTrials = 30;

for k = 1:K
    n = N(k);

    % generate random problem instance
    B = randn(n,p);
    c = randn(p);
    C = c + c.';
    x = randn(n,1);
    y = randn(p,1);
    f = my_dct(x) + B*y;
    g = B'*x + C*y;

    T = 0;
    for j = 1:nTrials
        t0 = tic;

        % SOLVE for x and y here
        z = g-B'*my_idct(f);
        y_hat = (C - B'*my_idct(B))\z;
        x_hat = my_idct( f - B*y_hat);

        t = toc(t0);
        T = T+t;
    end

    exec_time(k) = T/nTrials;
end

%% Plot Results

plot(N, exec_time, 'o-','linewidth',2)
yh = ylabel('Execution time (s)');
xh = xlabel('Problem size n');
```

```
set(gca,'box','off','tickdir','out')
set([gca xh yh],'fontsize',14)
```

As expected, the execution time is roughly linear in n: