EE364a: Convex Optimization I S. Boyd
March 14–15 or March 15–16, 2014

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 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 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. 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. Some are easy. Others, not so much.

Be sure you are using the most recent version of CVX, which is Version 2.1, build 1074. You can check this using the command `cvx_version`. Check your email often during the exam, just in case we need to send out an important announcement.

Some problems involve applications. But you do not need to know *anything* about the problem area to solve the problem; the problem statement contains everything you need.

1. *Lightest structure that resists a set of loads.* We consider a mechanical structure in 2D (for simplicity) which consists of a set of m nodes, with known positions $p_1, \ldots, p_m \in \mathbf{R}^2$, connected by a set of n bars (also called struts or elements), with cross-sectional areas $a_1, \ldots, a_n \in \mathbf{R}_+$, and internal tensions $t_1, \ldots, t_n \in \mathbf{R}$.

   Bar j is connected between nodes r_j and s_j. (The indices $r_1, \ldots, r_n$ and $s_1, \ldots, s_n$ give the structure topology.) The length of bar j is $L_j = \|p_{r_j} - p_{s_j}\|_2$, and the total volume of the bars is $V = \sum_{j=1}^n a_j L_j$. (The total weight is proportional to the total volume.)

   Bar j applies a force $(t_j/L_j)(p_{r_j} - p_{s_j}) \in \mathbf{R}^2$ to node s_j and the negative of this force to node r_j. Thus, positive tension in a bar pulls its two adjacent nodes towards each other; negative tension (also called compression) pushes them apart. The ratio of the tension in a bar to its cross-sectional area is limited by its yield strength, which is symmetric in tension and compression: $|t_j| \leq \sigma a_j$, where $\sigma > 0$ is a known constant that depends on the material.

   The nodes are divided into two groups: free and fixed. We will take nodes $1, \ldots, k$ to be free, and nodes $k+1, \ldots, m$ to be fixed. Roughly speaking, the fixed nodes are firmly attached to the ground, or a rigid structure connected to the ground; the free ones are not.

   A *loading* consists of a set of external forces, $f_1, \ldots, f_k \in \mathbf{R}^2$ applied to the free nodes. Each free node must be in equilibrium, which means that the sum of the forces applied to it by the bars and the external force is zero. The structure can *resist* a loading (without collapsing) if there exists a set of bar tensions that satisfy the tension bounds and force equilibrium constraints. (For those with knowledge of statics, these conditions correspond to a structure made entirely with pin joints.)

   Finally, we get to the problem. You are given a set of M loadings, *i.e.*, $f_1^{(i)}, \ldots, f_k^{(i)} \in \mathbf{R}^2$, $i = 1, \ldots, M$. The goal is to find the bar cross-sectional areas that minimize the structure volume V while resisting all of the given loadings. (Thus, you are to find *one* set of bar cross-sectional areas, and M sets of tensions.) Using the problem data provided in `lightest_struct_data.m`, report $V^\star$ and V^{unif}, the smallest feasible structure volume when all bars have the same cross-sectional area. The node positions are given as a $2 \times m$ matrix `P`, and the loadings as a $2 \times k \times M$ array `F`. Use the code included in the data file to visualize the structure with the bar cross-sectional areas that you find, and provide the plot in your solution.

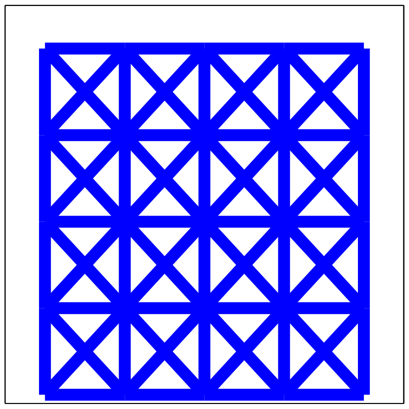
   Hint. You might find the graph incidence matrix $A \in \mathbf{R}^{m \times n}$ useful. It is defined as

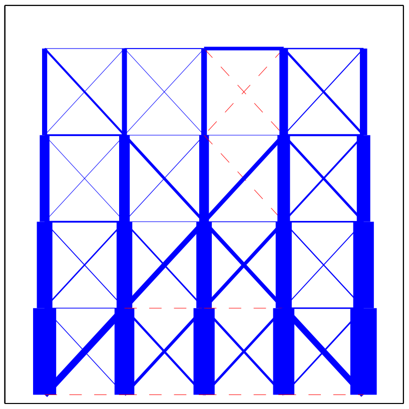
$$A_{ij} = \begin{cases} +1 & i = r_j \\ -1 & i = s_j \\ 0 & \text{otherwise.} \end{cases}$$

   Remark. You could reasonably ask, 'Does a mechanical structure really solve a convex optimization problem to determine whether it should collapse?'. It sounds odd, but the answer is, yes it does.

Solution. We can use the graph incidence matrix A to express the force exerted by the bars on each node. For a loading $f_1, \dots, f_k$, define $G \in \mathbf{R}^{2\times m}$ such that g_i, the ith column of G, is the sum of the forces from each of the bars connected to node i. Then $G = -PADA^T$, where $P \in \mathbf{R}^{2\times m}$ is a matrix whose ith column is p_i, and $D \in \mathbf{R}^{n\times n}$ is a diagonal matrix such that $D_{jj} = t_j/L_j$. (To see this, note that the ith column of PAD is $(t_i/L_i)(p_{r_i} - p_{s_i})$, the force that bar i applies to the adjacent node r_i.) The force equilibrium constraints for the loading can then be written as $g_i + f_i = 0$, for $i = 1, \dots, k$. A single set of bar cross-sectional areas must satisfy the equilibrium constraints for each of the M loadings. The remaining constraints are easily formulated.

We find that $V^\star = 188.55$, and $V^{\text{unif}} = 492.00$.





The following code solves the problem.

```
% lightest structure that resists a set of loads
lightest_struct_data;

% form incidence matrix
A = zeros(m, n);
for i = 1:n
    A(r(i), i) = +1;
    A(s(i), i) = -1;
end

L = norms(P*A)';

% solve with all bars having same cross-sectional area
cvx_begin quiet
    variables a(n) t(n, M)
    expression G(2, m, M) % force due to bars
    minimize (a'*L)
    subject to
        a == mean(a);
        for i = 1:M
            abs(t(:, i)) <= sigma.*a;
            G(:, :, i) = -P*A*diag(t(:, i)./L)*A';
            G(:, 1:k, i) + F(:, :, i) == 0;
        end
cvx_end
fprintf('V^unif = %f\n', cvx_optval);

% plot
clf;
subplot(1,2,1); hold on;
for i = 1:n
    p1 = r(i); p2 = s(i);
    plt_str = 'b-';
    if a(i) < 0.001
        plt_str = 'r--';
    end
    plot([P(1, p1) P(1, p2)], [P(2, p1) P(2, p2)], ...
         plt_str, 'LineWidth', a(i));
end
axis([-0.5 N-0.5 -0.1 N-0.5]); axis square; box on;
set(gca, 'xtick', [], 'ytick', []);
hold off;

% solve with bars having different cross-sectional areas
cvx_begin quiet
    variables a(n) t(n, M)
    expression G(2, m, M) % force due to bars
    minimize (a'*L)
    subject to
        for i = 1:M
            abs(t(:, i)) <= sigma.*a;
            G(:, :, i) = -P*A*diag(t(:, i)./L)*A';
            G(:, 1:k, i) + F(:, :, i) == 0;
        end
cvx_end
fprintf('V^star = %f\n', cvx_optval);

% plot
subplot(1,2,2); hold on;
```

```
for i = 1:n
    p1 = r(i); p2 = s(i);
    plt_str = 'b-';
    width = a(i);
    if a(i) < 0.001
        plt_str = 'r--';
        width = 1;
    end
    plot([P(1, p1) P(1, p2)], [P(2, p1) P(2, p2)], ...
         plt_str, 'LineWidth', width);
end
axis([-0.5 N-0.5 -0.1 N-0.5]); axis square; box on;
set(gca, 'xtick', [], 'ytick', []);
hold off;

print -depsc lightest_struct.eps;
```

2. *Theory-applications split in a course.* A professor teaches an advanced course with 20 lectures, labeled $i = 1, \ldots, 20$. The course involves some interesting theoretical topics, and many practical applications of the theory. The professor must decide how to split each lecture between theory and applications. Let T_i and A_i denote the fraction of the ith lecture devoted to theory and applications, for $i = 1, \ldots, 20$. (We have $T_i \geq 0$, $A_i \geq 0$, and $T_i + A_i = 1$.)

   A certain amount of theory has to be covered before the applications can be taught. We model this in a crude way as

$$A_1 + \cdots + A_i \leq \phi(T_1 + \cdots + T_i), \quad i = 1, \ldots, 20,$$

   where $\phi : \mathbf{R} \to \mathbf{R}$ is a given nondecreasing function. We interpret $\phi(u)$ as the cumulative amount of applications that can be covered, when the cumulative amount of theory covered is u. We will use the simple form $\phi(u) = a(u-b)_+$, with $a, b > 0$, which means that no applications can be covered until b lectures of the theory is covered; after that, each lecture of theory covered opens the possibility of covering a lectures on applications.

   The theory-applications split affects the emotional state of students differently. We let s_i denote the emotional state of a student after lecture i, with $s_i = 0$ meaning neutral, $s_i > 0$ meaning happy, and $s_i < 0$ meaning unhappy. Careful studies have shown that s_i evolves via a linear recursion (dynamics)

$$s_i = (1-\theta)s_{i-1} + \theta(\alpha T_i + \beta A_i), \quad i = 1, \ldots, 20,$$

   with $s_0 = 0$. Here α and β are parameters (naturally interpreted as how much the student likes or dislikes theory and applications, respectively), and $\theta \in [0, 1]$ gives the emotional volatility of the student (*i.e.*, how quickly he or she reacts to the content of recent lectures). The student's terminal emotional state is s_{20}.

   Now consider a specific instance of the problem, with course material parameters $a = 2$, $b = 3$, and three groups of students, with emotional dynamics parameters given as follows.

	Group 1	Group 2	Group 3
θ	0.05	0.1	0.3
α	-0.1	0.8	-0.3
β	1.4	-0.3	0.7

   Find (four different) theory-applications splits that maximize the terminal emotional state of the first group, the terminal emotional state of the second group, the terminal emotional state of the third group, and, finally, the minimum of the terminal emotional states of all three groups.

   For each case, plot T_i and the emotional state s_i for the three groups, versus i. Report the numerical values of the terminal emotional states for each group, for each of the four theory-applications splits.

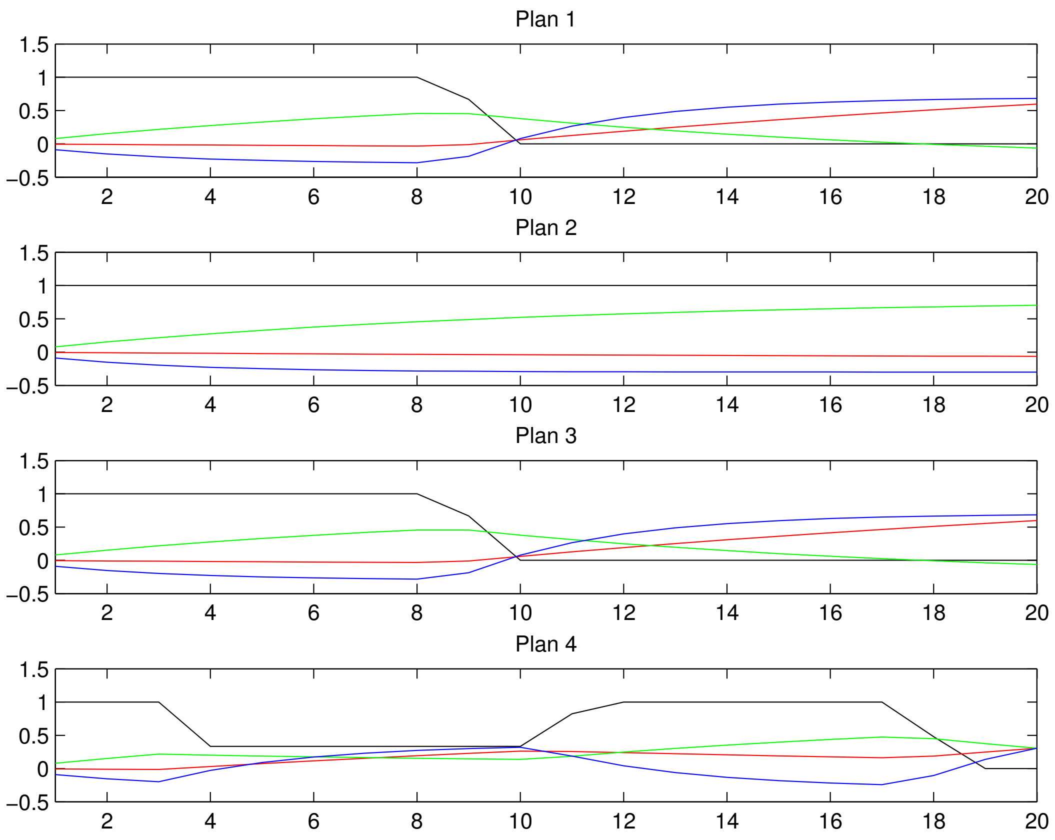
Solution. Because of the way that ϕ was chosen, the first b lectures have to be theory only, *i.e.*, $T_i = 1$, $A_i = 0$ for $i = 1, \ldots, b$. Using this observation, we can rewrite the condition on the theory-applications split:

$$A_{b+1} + \cdots + A_i \leq a(T_{b+1} + \cdots + T_i), \quad i = b+1, \ldots, 20.$$

This is a linear inequality in the variables $T_{b+1}, \ldots, T_{20}$, and $A_{b+1}, \ldots, A_{20}$.

Note that each s_i is also linear in the same set of variables. Thus, maximizing the given objective functions is a linear program.

Solving the numerical instance gives the following plots. The black curve shows T_i. The red, green, and blue curves show the emotional states of the three student groups.



The terminal emotional states for the three groups under the four different lecture plans are given by the table below.

	Group 1	Group 2	Group 3
Plan 1	0.597	-0.064	0.682
Plan 2	-0.064	0.703	-0.300
Plan 3	0.597	-0.064	0.682
Plan 4	0.306	0.306	0.306

The following code solves the problem and generates the plots shown above.

```
% theory-applications split in a course
clear all; clf;

% set up the parameters
a = 2; b = 3;
theta = [ 0.05; 0.1;  0.3];
alpha = [-0.1;  0.8; -0.3];
beta  = [ 1.4; -0.3;  0.7];
n = 20; % number of lectures
m = 3; % number of student groups

for plan = 1:m+1
    cvx_begin quiet
        variable T(n)
        expressions s(m, n+1) obj

        % compute the emotional states
        for i = 1:n
            s(:, i+1) = (1-theta).*s(:, i) ...
                        + theta.*(alpha*T(i)+beta*(1-T(i)));
        end

        if plan == 4
            obj = min(s(:, n+1));
        else
            obj = s(plan, n+1);
        end

        maximize obj
        subject to
            T >= 0;
            T <= 1;
            T(1:b) == 1;
            cumsum(1-T(b+1:n)) <= a*cumsum(T(b+1:n));
    cvx_end

    % plot
    subplot(4, 1, plan);
    plot(1:n, T, 'k', ...
         1:n, s(1, 2:n+1), 'r', ...
         1:n, s(2, 2:n+1), 'g', ...
         1:n, s(3, 2:n+1), 'b');
    title(sprintf('Plan %d', plan));
    axis([1 n -0.5 1.5]);
    fprintf('Plan %d: %f %f %f\n', plan, ...
            s(1, n+1), s(2, n+1), s(3, n+1));
end

print -depsc theory_appls.eps;
```

3. *Optimal electric motor drive currents.* In this problem you will design the drive current waveforms for an AC (alternating current) electric motor. The motor has a magnetic rotor which spins with constant angular velocity $\omega \geq 0$ inside the stationary stator. The stator contains three circuits (called *phase windings*) with (vector) current waveform $i : \mathbf{R} \to \mathbf{R}^3$ and (vector) voltage waveform $v : \mathbf{R} \to \mathbf{R}^3$, which are 2π-periodic functions of the angular position θ of the rotor. The circuit dynamics are

$$v(\theta) = Ri(\theta) + \omega L \frac{d}{d\theta} i(\theta) + \omega k(\theta),$$

where $R \in \mathbf{S}^3_{++}$ is the resistance matrix, $L \in \mathbf{S}^3_{++}$ is the inductance matrix, and $k : \mathbf{R} \to \mathbf{R}^3$, a 2π-periodic function of θ, is the back-EMF waveform (which encodes the electromagnetic coupling between the rotor permanent magnets and the phase windings). The angular velocity ω, the matrices R and L, and the back-EMF waveform k, are known.

We must have $|v_i(\theta)| \leq v^{\text{supply}}$, $i = 1, 2, 3$, where v^{supply} is the (given) supply voltage. The output torque of the motor at rotor position θ is $\tau(\theta) = k(\theta)^T i(\theta)$. We will require the torque to have a given constant nonnegative value: $\tau(\theta) = \tau^{\text{des}}$ for all θ.

The average power loss in the motor is

$$P^{\text{loss}} = \frac{1}{2\pi} \int_0^{2\pi} i(\theta)^T R i(\theta) \, d\theta.$$

The mechanical output power is $P^{\text{out}} = \omega \tau^{\text{des}}$, and the motor efficiency is

$$\eta = P^{\text{out}} / (P^{\text{out}} + P^{\text{loss}}).$$

The objective is to choose the current and voltage waveforms to maximize η.

Discretization. To solve this problem we consider a discretized version in which θ takes on the N values $\theta = h, 2h, \ldots, Nh$, where $h = 2\pi/N$. We impose the voltage and torque constraints for these values of θ. We approximate the power loss as

$$P^{\text{loss}} = (1/N) \textstyle\sum_{j=1}^N i(jh)^T R i(jh).$$

The circuit dynamics are approximated as

$$v(jh) = Ri(jh) + \omega L \frac{i((j+1)h) - i(jh)}{h} + \omega k(jh), \quad j = 1, \ldots, N,$$

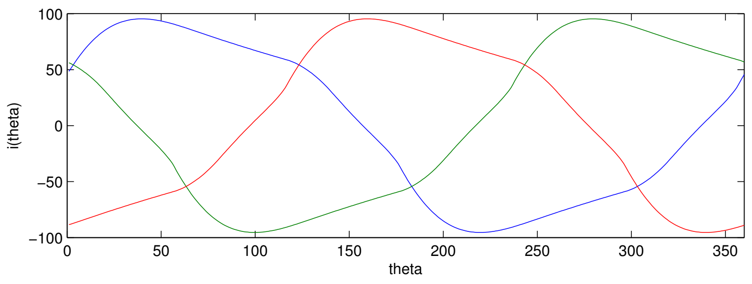
where here we take $i((N+1)h) = i(h)$ (by periodicity).

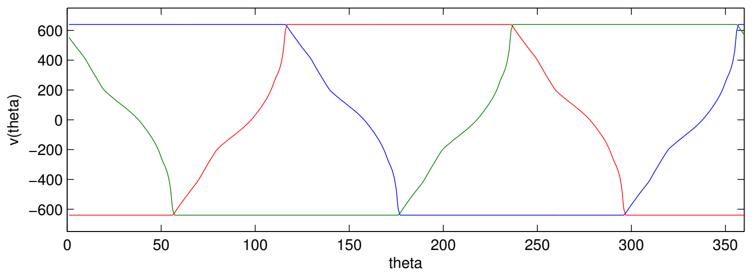
Find optimal (discretized) current and voltage waveforms for the problem instance with data given in `ac_motor_data.m`. The back-EMF waveform is given as a $3 \times N$ matrix `K`. Plot the three current waveform components on one plot, and the three voltage waveforms on another. Give the efficiency obtained.

Solution. We first note that all of the constraints are convex as stated. To maximize the efficiency, we minimize the power loss, which is a sum of convex quadratic terms. The following code will solve the problem.

```
% optimal electric motor drive currents
ac_motor_data;

Rhalf = chol(R);
cvx_begin
    variables I(3, N) V(3, N)
    minimize (norm(Rhalf*I, 'fro'))  % sqrt(N*Ploss)
    subject to
        V == R*I + omega*K + omega*L*(I(:, [2:N, 1])-I)/h; % dynamics
        tau_des == sum(K.*I); % torque constraint
        abs(V) <= V_supply; % voltage limits
cvx_end

Ploss = cvx_optval^2/N;
Pout = omega*tau_des;
eta = Pout/(Pout+Ploss);

fprintf('Maximum efficiency: %f\n', eta);

% plot
subplot(2, 1, 1);
plot(I(:, 1:N)');
xlabel('theta'); ylabel('i(theta)');
axis([0, 360, -100, 100]);

subplot(2, 1, 2);
plot(V');
xlabel('theta'); ylabel('v(theta)');
axis([0, 360, -750, 750]);

print -depsc ac_motor.eps;
```

The maximum efficiency is 90.2%. The optimal current and voltage waveforms are shown below.


i(theta)
100
50
0
−50
−100
0
50
100
150
200
250
300
350
theta


v(theta)
600
400
200
0
−200
−400
−600
0
50
100
150
200
250
300
350
theta

4. *Multi-label support vector machine.* The basic SVM described in the book is used for classification of data with two labels. In this problem we explore an extension of SVM that can be used to carry out classification of data with more than two labels. Our data consists of pairs $(x_i, y_i) \in \mathbf{R}^n \times \{1, \ldots, K\}$, $i = 1, \ldots, m$, where x_i is the feature vector and y_i is the label of the ith data point. (So the labels can take the values $1, \ldots, K$.) Our classifier will use K affine functions, $f_k(x) = a_k^T x + b_k$, $k = 1, \ldots, K$, which we also collect into affine function from $\mathbf{R}^n$ into $\mathbf{R}^K$ as $f(x) = Ax + b$. (The rows of A are a_k^T.) Given feature vector x, we guess the label $\hat{y} = \mathrm{argmax}_k\, f_k(x)$. We assume that exact ties never occur, or if they do, an arbitrary choice can be made. Note that if a multiple of $\mathbf{1}$ is added to b, the classifier does not change. Thus, without loss of generality, we can assume that $\mathbf{1}^T b = 0$.

   To correctly classify the data examples, we need $f_{y_i}(x_i) > \max_{k \neq y_i} f_k(x_i)$ for all i. This is a set of homogeneous strict inequalities in a_k and b_k, which are feasible if and only if the set of nonstrict inequalities $f_{y_i}(x_i) \geq 1 + \max_{k \neq y_i} f_k(x_i)$ are feasible. This motivates the loss function

   $$L(A, b) = \sum_{i=1}^{m} \left(1 + \max_{k \neq y_i} f_k(x_i) - f_{y_i}(x_i) \right)_+ ,$$

   where $(u)_+ = \max\{u, 0\}$. The multi-label SVM chooses A and b to minimize

   $$L(A, b) + \mu \|A\|_F^2,$$

   subject to $\mathbf{1}^T b = 0$, where $\mu > 0$ is a regularization parameter. (Several variations on this are possible, such as regularizing b as well, or replacing the Frobenius norm squared with the sum of norms of the columns of A.)

   (a) Show how to find A and b using convex optimization. Be sure to justify any changes of variables or reformulation (if needed), and convexity of the objective and constraints in your formulation.

   (b) Carry out multi-label SVM on the data given in `multi_label_svm_data.m`. Use the data given in `X` and `y` to fit the SVM model, for a range of values of μ. This data set includes an additional set of data, `Xtest` and `ytest`, that you can use to test the SVM models. Plot the test set classification error rate (*i.e.*, the fraction of data examples in the test set for which $\hat{y} \neq y$) versus μ.

   You don't need to try more than 10 or 20 values of μ, and we suggest choosing them uniformly on a log scale, from (say) 10^{-2} to 10^2.

   Solution.

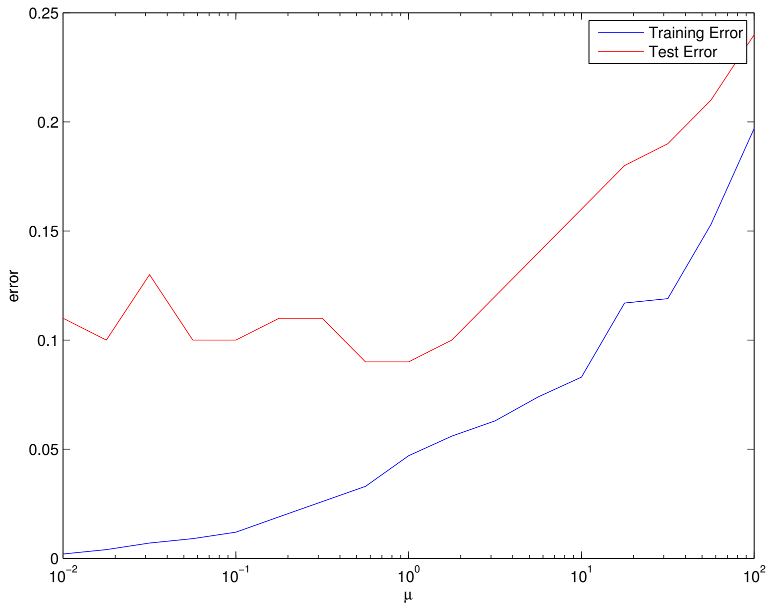
   (a) The multi-label SVM problem is convexas stated. The variables are A and b. The only constraint, $\mathbf{1}^T b = 0$, is linear. The regularization term in the objective,

$\mu\|A\|_F^2$, is convex quadratic. Let us justify that the loss function term $L(A,b)$ is convex, which we can do by showing each term,

$$\left(1 + \max_{k \neq y_i} f_k(x_i) - f_{y_i}(x_i)\right)_+,$$

is convex. Since $f_k(x_i)$ is linear in the variables, $\max_{k\neq y_i} f_k(x_i)$ is convex. The argument of $(\cdot)_+$ is a sum of a constant, a convex, and a linear function, and so is convex. The function $(\cdot)_+$ is convex and nondecreasing, so by the composition rule, the term above is convex.

(b) The plot below shows the training and test error obtained for different values of μ. A reasonable value to choose would be around $\mu = 1$, although smaller values seem to work well too.



The following code solves the problem.

```
% multi-label support vector machine
multi_label_svm_data;

mus = 10.^(-2:0.25:2);
errorTrain = [];
errorTest  = [];

for mu = mus
    cvx_begin quiet
        variables A(K, n) b(K)
        expressions L f(K, mTrain)
        % f(k, i) stores f_k(x_i)
        f = A*x + b*ones([1 mTrain]);
        L = 0;
        for k = 1:K
            % process all examples with y_i = k simultaneously
            ind = [1:k-1 k+1:K];
            L = L + sum(pos(1 + max(f(ind, y==k), [], 1) ...
                            - f(k, y==k)));
        end
        minimize (L + mu*sum(sum(A.^2)))
        subject to
            sum(b) == 0;
    cvx_end

    [~, indTrain] = max(A*x + b*ones([1 mTrain]), [], 1);
    errorTrain(end+1) = sum(indTrain~=y)/mTrain;
    [~, indTest] = max(A*xtest + b*ones([1 mTest]), [], 1);
    errorTest(end+1) = sum(indTest~=ytest)/mTest;
end

% plot
clf;
semilogx(mus, errorTrain); hold on;
semilogx(mus, errorTest, 'r'); hold off;
xlabel('\mu'); ylabel('error');
legend('Training Error', 'Test Error');
print -depsc multi_label_svm.eps;
```

5. *De-leveraging.* We consider a multi-period portfolio optimization problem, with n assets and T time periods, where $x_t \in \mathbf{R}^n$ gives the holdings (say, in dollars) at time t, with negative entries denoting, as usual, short positions. For each time period the return vector has mean $\mu \in \mathbf{R}^n$ and covariance $\Sigma \in \mathbf{S}^n_{++}$. (These are known.)

   The initial portfolio x_0 maximizes the risk-adjusted expected return $\mu^T x - \gamma x^T \Sigma x$, where $\gamma > 0$, subject to the leverage limit constraint $\|x\|_1 \leq L^{\rm init}$, where $L^{\rm init} > 0$ is the given initial leverage limit. (There are several different ways to measure leverage; here we use the sum of the total short and long positions.) The final portfolio x_T maximizes the risk-adjusted return, subject to $\|x\|_1 \leq L^{\rm new}$, where $L^{\rm new} > 0$ is the given final leverage limit (with $L^{\rm new} < L^{\rm init}$). This uniquely determines x_0 and x_T, since the objective is strictly concave.

   The question is how to move from x_0 to x_T, *i.e.*, how to choose $x_1, \ldots, x_{T-1}$. We will do this so as to maximize the objective

$$J = \sum_{t=1}^{T} \left(\mu^T x_t - \gamma x_t^T \Sigma x_t - \phi(x_t - x_{t-1}) \right),$$

   which is the total risk-adjusted expected return, minus the total transaction cost. The transaction cost function ϕ has the form

$$\phi(u) = \sum_{i=1}^{n} \left(\kappa_i |u_i| + \lambda_i u_i^2 \right),$$

   where $\kappa \succeq 0$ and $\lambda \succeq 0$ are known parameters. We will require that $\|x_t\|_1 \leq L^{\rm init}$, for $t = 1, \ldots, T-1$. In other words, the leverage limit is the initial leverage limit up until the deadline T, when it drops to the new lower value.

   (a) Explain how to find the portfolio sequence $x_1^\star, \ldots, x_{T-1}^\star$ that maximizes J subject to the leverage limit constraints.

   (b) Find the optimal portfolio sequence $x_t^\star$ for the problem instance with data given in `deleveraging_data.m`. Compare this sequence with two others: $x_t^{\rm lp} = x_0$ for $t = 1, \ldots, T-1$ (*i.e.*, one that does all trading at the last possible period), and the linearly interpolated portfolio sequence

$$x_t^{\rm lin} = (1 - t/T)x_0 + (t/T)x_T, \quad t = 1, \ldots, T-1.$$

   For each of these three portfolio sequences, give the objective value obtained, and plot the risk and transaction cost adjusted return,

$$\mu^T x_t - \gamma x_t^T \Sigma x_t - \phi(x_t - x_{t-1}),$$

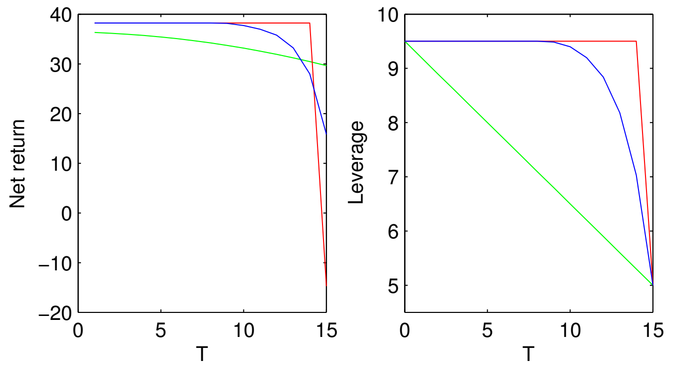
   and the leverage $\|x_t\|_1$, versus t, for $t = 0, \ldots, T$. Also, for each of the three portfolio sequences, generate a single plot that shows how the holdings $(x_t)_i$ of the n assets change over time, for $i = 1, \ldots, n$.

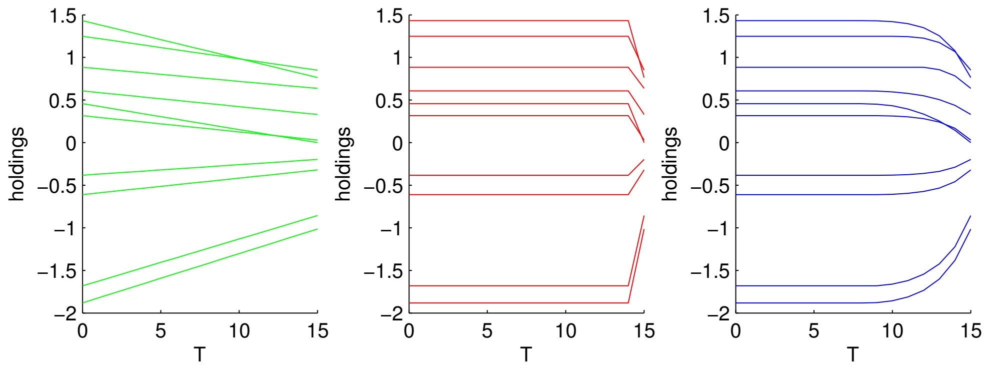
   Give a *very short* (one or two sentence) intuitive explanation of the results.

Solution.

(a) It is easy to see that ϕ is convex in u. By the composition rule, it follows that each period cost at time t is concave in $x_1, \dots, x_{T-1}$. Then, maximizing J subject to the leverage limit constraints is a convex optimization problem. The constraints for this optimization problem are straightforward: $\|x_i\|_1 \leq L^{\text{init}}, i = 1, ..., T-1$, $x_0 = x_0^\star$ and $x_T = x_T^\star$, where $x_0^\star$ and $x_T^\star$ are both given by maximizing the risk adjusted return (without transaction cost) subject to initial and last leverage limit.

(b) The total net returns for the linearly interpolated portfolio sequence, the portfolio sequence that does all the trading at the last period, and the optimal sequence are 506.18, 520.42 and 531.38 respectively. In the following graphs, the three sequences are drawn in green, red, and blue, in this order.





Intuitive explanation. The linearly interpolated portfolio smoothly changes its positions, so it doesn't incur much transaction cost. On the other hand, it starts deleveraging well before the deadline, so it loses some opportunity to improve the objective in the early steps. When we make all trades in the last period, we get a good objective value up until then, but then we pay for it in transaction costs in the last period, and the total objective suffers. When you transition just right, you combine the best of both strategies: you keep the portfolio near the optimal one for high leverage for about 10 periods, then smoothly transition to the new portfolio, in such a way that the total objective is maximized.

The following code solve the problem.

```
% de-leveraging
deleveraging_data;

% compute initial and final portfolio
cvx_begin quiet
    variable x0_star(n)
    maximize (mu'*x0_star - gamma*quad_form(x0_star, Sigma))
    subject to
        norm(x0_star, 1) <= Linit;
cvx_end

cvx_begin quiet
    variable xT_star(n)
    maximize (mu'*xT_star - gamma*quad_form(xT_star, Sigma))
    subject to
        norm(xT_star, 1) <= Lnew;
cvx_end

% linear interpolation
Xlin = zeros(n, T+1);
for i = 1:T+1
    Xlin(:, i) = (1-(i-1)/T)*x0_star + (i-1)/T*xT_star;
end

% trading at the last period
Xlp = [repmat(x0_star, 1, T) xT_star];

% optimization
cvx_begin quiet
    variable X(n, T+1)
    expressions dX(n, T) ret trs
    dX = X(:, 2:T+1) - X(:, 1:T);
    ret = 0; trs = 0;
    for t = 2:T+1
        ret = ret + mu'*X(:, t) - gamma*quad_form(X(:, t), Sigma);
        trs = trs + kappa'*abs(dX(:, t-1)) ...
              + quad_form(dX(:, t-1), diag(lambda));
    end
    maximize ret - trs
    subject to
        X(:, 1) == x0_star;
        X(:, T+1) == xT_star;
        norms(X(:, 2:T), 1) <= Linit;
cvx_end

levOpt = norms(X, 1);
levLin = norms(Xlin, 1);
levLp  = norms(Xlp, 1);

retOpt = X'*mu - gamma*diag(X'*Sigma*X);
retLin = Xlin'*mu - gamma*diag(Xlin'*Sigma*Xlin);
retLp  = Xlp'*mu - gamma*diag(Xlp'*Sigma*Xlp);

dXlin = Xlin(:, 2:T+1) - Xlin(:, 1:T);
dXlp  = Xlp(:, 2:T+1) - Xlp(:, 1:T);

trsOpt = abs(dX)'*kappa + diag(dX'*diag(lambda)*dX);
trsLin = abs(dXlin)'*kappa + diag(dXlin'*diag(lambda)*dXlin);
trsLp  = abs(dXlp)'*kappa + diag(dXlp'*diag(lambda)*dXlp);

netReturnOpt = retOpt(2:T+1) - trsOpt;
netReturnLin = retLin(2:T+1) - trsLin;
netReturnLp  = retLp (2:T+1) - trsLp;

fprintf('J of x^lin : %f\n', sum(netReturnLin));
fprintf('J of x^lp  : %f\n', sum(netReturnLp));
fprintf('J of x^star: %f\n', sum(netReturnOpt));

% plot
clf;
subplot(2, 3, 1.5);
plot(1:T, netReturnLin, 'g', 1:T, netReturnLp, 'r', ...
     1:T, netReturnOpt, 'b');
xlabel('T'); ylabel('Net return'); xlim([0, T]);

subplot(2, 3, 2.5);
plot(0:T, levLin, 'g', 0:T, levLp, 'r', 0:T, levOpt, 'b');
xlabel('T'); ylabel('Leverage'); axis([0 T Lnew-0.5 Linit+0.5]);

subplot(2, 3, 4); % x^lin
```

```
hold on;
for i = 1:n
    plot(0:T, Xlin(i, :), 'g');
end
hold off;
xlabel('T'); ylabel('holdings'); xlim([0 T]);

subplot(2, 3, 5); % x^lp
hold on;
for i = 1:n
    plot(0:T, Xlp(i, :), 'r');
end
hold off;
xlabel('T'); ylabel('holdings'); xlim([0 T]);

subplot(2, 3, 6); % x^star
hold on;
for i = 1:n
    plot(0:T, X(i, :), 'b');
end
hold off;
xlabel('T'); ylabel('holdings'); xlim([0 T]);

print -depsc deleveraging.eps;
```

6. *Worst-case variance.* Suppose Z is a random variable on $\mathbf{R}^n$ with covariance matrix $\Sigma \in \mathbf{S}^n_+$. Let $c \in \mathbf{R}^n$. The variance of $Y = c^T Z$ is $\mathbf{var}(Y) = c^T \Sigma c$. We define the *worst-case variance* of Y, denoted $\mathbf{wcvar}(Y)$, as the maximum possible value of $c^T \tilde{\Sigma} c$, over all $\tilde{\Sigma} \in \mathbf{S}^n_+$ that satisfy $\Sigma_{ii} = \tilde{\Sigma}_{ii}$, $i = 1, \ldots, n$. In other words, the worst-case variance of Y is the maximum possible variance, if we are allowed to arbitrarily change the correlations between Z_i and Z_j. Of course we have $\mathbf{wcvar}(Y) \geq \mathbf{var}(Y)$.

   (a) Find a simple expression for $\mathbf{wcvar}(Y)$ in terms of c and the diagonal entries of Σ. You must justify your expression.

   (b) *Portfolio optimization.* Explain how to find the portfolio $x \in \mathbf{R}^n$ that maximizes the expected return $\mu^T x$ subject to a limit on risk, $\mathbf{var}(r^T x) = x^T \Sigma x \leq R$, and a limit on worst-case risk $\mathbf{wcvar}(r^T x) \leq R^{\text{wc}}$, where $R > 0$ and $R^{\text{wc}} > R$ are given. Here $\mu = \mathbf{E}\, r$ and $\Sigma = \mathbf{E}(r - \mu)(r - \mu)^T$ are the (given) mean and covariance of the (random) return vector $r \in \mathbf{R}^n$.

   (c) Carry out the method of part (b) for the problem instance with data given in `wc_risk_portfolio_opt_data.m`. Also find the optimal portfolio when the worst-case risk limit is ignored. Find the expected return and worst-case risk for these two portfolios.

Remark. If a portfolio is highly leveraged, and the correlations in the returns change drastically, you (the portfolio manager) can be in big trouble, since you are now exposed to much more risk than you thought you were. And yes, this (almost exactly) has happened.

Solution.

(a) We will show that

$$\mathbf{wcvar}(Y) = \left(\sum_{i=1}^n |c_i| \sqrt{\Sigma_{ii}} \right)^2.$$

This worst-case variance value is obtained by the matrix $\Sigma^{\text{wc}} = dd^T$, where $d \in \mathbf{R}^n$ is defined as

$$d_i = \text{sign}(c_i) \sqrt{\Sigma_{ii}}, \quad i = 1, \ldots, n.$$

Note that $\Sigma^{\text{wc}} \in \mathbf{S}^n_+$ and its diagonal entries are the same as Σ.

To show this (and also, to derive it) we proceed as follows. For any $\tilde{\Sigma} \in \mathbf{S}^n_+$ that satisfies $\Sigma_{ii} = \tilde{\Sigma}_{ii}$ for all i, we have

$$\begin{aligned} c^T \tilde{\Sigma} c &= \sum_{i=1}^n c_i^2 \tilde{\Sigma}_{ii} + \sum_{i \neq j} c_i c_j \tilde{\Sigma}_{ij} \\ &= \sum_{i=1}^n c_i^2 \Sigma_{ii} + \sum_{i \neq j} c_i c_j \tilde{\Sigma}_{ij}. \end{aligned}$$

Since $\tilde{\Sigma} \succeq 0$, we must have

$$|\tilde{\Sigma}_{ij}| \leq \sqrt{\tilde{\Sigma}_{ii}\tilde{\Sigma}_{jj}} = \sqrt{\Sigma_{ii}\Sigma_{jj}}$$

for $i \neq j$. If we maximize $c^T\tilde{\Sigma}c$ over the off-diagonal elements of $\tilde{\Sigma}$ subject to this limit, we find that the maximum occurs when

$$\tilde{\Sigma}_{ij} = \mathbf{sign}(c_i c_j)\sqrt{\Sigma_{ii}\Sigma_{jj}}$$

for $i \neq j$. We can re-write this as

$$\tilde{\Sigma}_{ij} = \sqrt{\Sigma_{ii}\Sigma_{jj}}\,\mathbf{sign}(c_i)\,\mathbf{sign}(c_j), \quad i, j = 1, \dots, n.$$

So far, we have not taken into account the constraint that $\tilde{\Sigma} \succeq 0$. Fortunately, the matrix above satisfies this, since we can express it as $\tilde{\Sigma} = dd^T$.

(b) Given the data μ, Σ, R, and $R^{\rm wc}$, we solve the optimization problem

$$\begin{array}{ll} \text{maximize} & \mu^T x \\ \text{subject to} & x^T\Sigma x \leq R \\ & \sum_{i=1}^n |x_i|\sqrt{\Sigma_{ii}} \leq \sqrt{R^{\rm wc}}. \end{array}$$

(c) The optimal portfolio with the worst-case risk limit is

$$x^\star = (2.263,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0.698,\ -1.673,\ 0),$$

with expected return $\mu^T x^\star = 8.195$, and worst case risk $\mathbf{wcvar}(r^T x^\star) = 10$.

The optimal portfolio with the worst-case risk limit ignored is

$$x^\star = (1.625,\ 1.214,\ -0.672,\ -1.821,\ 2.395,\ -0.892,\ 0.533,\ 3.967,\ -0.565,\ -2.769),$$

with expected return $\mu^T x^\star = 13.512$, and worst case risk $\mathbf{wcvar}(r^T x^\star) = 117.197$.

The following code solves the problem.

```
% worst-case variance
wc_risk_portfolio_opt_data;

cvx_begin quiet
    variable x(n)
    maximize (mu'*x)
    subject to
        quad_form(x, Sigma) <= R;
        norm(sqrt(diag(Sigma)).*x, 1) <= sqrt(R_wc);
cvx_end

fprintf('Portfolio with the worst-case risk limit:\n')
fprintf('  Expected return: %.3f\n', mu'*x);
fprintf('  Variance: %.3f\n', quad_form(x, Sigma));
fprintf('  Worst case risk: %.3f\n', norm(sqrt(diag(Sigma)).*x, 1)^2);
fprintf('  x = (');
for i = 1:n
    fprintf('%.3f, ', x(i));
end
fprintf(')\n\n');

cvx_begin quiet
    variable x(n)
    maximize (mu'*x)
    subject to
        quad_form(x, Sigma) <= R;
cvx_end

fprintf('Portfolio without the worst-case risk limit:\n')
fprintf('  Expected return: %.3f\n', mu'*x);
fprintf('  Variance: %.3f\n', quad_form(x, Sigma));
fprintf('  Worst case risk: %.3f\n', norm(sqrt(diag(Sigma)).*x, 1)^2);
fprintf('  x = (');
for i = 1:n
    fprintf('%.3f, ', x(i));
end
fprintf(')\n');
```

7. *Smallest confidence ellipsoid.* Suppose the random variable X on $\mathbf{R}^n$ has log-concave density p. Formulate the following problem as a convex optimization problem: Find an ellipsoid $\mathcal{E}$ that satisfies $\mathbf{Prob}(X \in \mathcal{E}) \geq 0.95$ and is smallest, in the sense of minimizing the sum of the squares of its semi-axis lengths. You do not need to worry about how to solve the resulting convex optimization problem; it is enough to formulate the smallest confidence ellipsoid problem as the problem of minimizing a convex function over a convex set involving the parameters that define $\mathcal{E}$.

   Solution. We parametrize the ellipsoid as $\mathcal{E}(c, P) = \{x \mid (x-c)^T P^{-1}(x-c) \leq 1\}$, where $P \succ 0$. The semi-axis lengths are $\lambda_i^{1/2}$, where λ_i are the eigenvalues of P. The sum of the squares of the semi-axis lengths is $\sum_i \lambda_i = \mathbf{Tr}\, P$. So our job is to minimize $\mathbf{Tr}\, P$ subject to $\mathbf{Prob}(X \in \mathcal{E}) \geq 0.95$.

   Define $g(x, c, P)$ as the 0-1 indicator function of $\mathcal{E}$, *i.e.*,

   $$g(x, c, P) = \begin{cases} 1 & (x-c)^T P^{-1}(x-c) \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

   Since $(x-c)^T P^{-1}(x-c)$ is a convex function of (x, c, P) (for $P \succ 0$), $\log g$ is concave in (x, c, P), so g is log-concave in (x, c, P). By the integration rule for log-concave functions,

   $$\int p(x) g(x, c, P)\, dx = \mathbf{Prob}(X \in \mathcal{E}(c, P))$$

   is log-concave in (c, P). So for any η, $\mathbf{Prob}(X \in \mathcal{E}(c, P)) \geq \eta$ is a convex constraint in (c, P). In particular,

   $$\{(c, P) \mid \mathbf{Prob}(X \in \mathcal{E}(c, P)) \geq 0.95\}$$

   is a convex set. We simply minimize (the linear function) $\mathbf{Tr}\, P$ over this set to get the smallest confidence ellipsoid.

8. *Lyapunov analysis of a dynamical system.* We consider a discrete-time time-varying linear dynamical system with state $x_t \in \mathbf{R}^n$. The state propagates according to the linear recursion $x_{t+1} = A_t x_t$, for $t = 0, 1, \ldots$, where the matrices A_t are unknown but satisfy $A_t \in \mathcal{A} = \{A^{(1)}, \ldots, A^{(K)}\}$, where $A^{(1)}, \ldots, A^{(K)}$ are known. (In computer science, this would be called a non-deterministic linear automaton.) We call the sequence $x_0, x_1, \ldots$ a *trajectory* of the system. There are infinitely many trajectories, one for each sequence $A_0, A_1, \ldots$.

   The *Lyapunov exponent* κ of the system is defined as

$$\kappa = \sup_{A_0, A_1, \ldots} \limsup_{t \to \infty} \|x_t\|_2^{1/t}.$$

   (If you don't know what sup and lim sup mean, you can replace them with max and lim, respectively.) Roughly speaking, this means that all trajectories grow no faster than κ^t. When $\kappa < 1$, the system is called *exponentially stable.*

   It is a hard problem to determine the Lyapunov exponent of the system, or whether the system is exponentially stable, given the data $A^{(1)}, \ldots, A^{(K)}$. In this problem we explore a powerful method for computing an upper bound on the Lyapunov exponent.

   (a) Let $P \in \mathbf{S}^n_{++}$ and define $V(x) = x^T P x$. Suppose V satisfies

$$V(A^{(i)}x) \le \gamma^2 V(x) \text{ for all } x \in \mathbf{R}^n, \; i = 1, \ldots, K.$$

   Show that $\kappa \le \gamma$. Thus γ is an upper bound on the Lyapunov exponent κ. (The function V is called a quadratic *Lyapunov function* for the system.)

   (b) Explain how to use convex or quasiconvex optimization to find a matrix $P \in \mathbf{S}^n_{++}$ with the smallest value of γ, *i.e.*, with the best upper bound on κ. You must justify your formulation.

   (c) Carry out the method of part (b) for the specific problem with data given in `lyap_exp_bound_data.m`. Report the best upper bound on κ, to a tolerance of 0.01. The data $A^{(i)}$ are given as a cell array; `A{i}` gives $A^{(i)}$.

   (d) *Approximate worst-case trajectory simulation.* The quadratic Lyapunov function found in part (c) can be used to generate sequences of A_t that tend to result in large values of $\|x_t\|_2^{1/t}$. Start from a random vector x_0. At each t, generate x_{t+1} by choosing $A_t = A^{(i)}$ that maximizes $V(A^{(i)}x_t)$, where P is computed from part (c). Do this for 50 time steps, and generate 5 such trajectories. Plot $\|x_t\|_2^{1/t}$ and γ against t to verify that the bound you obtained in the previous part is valid. Report the lower bound on the Lyapunov exponent that the trajectories suggest.

Solution.

(a) Suppose V satisfies the conditions given above, and $x_0, x_1, \ldots$ is any trajectory of the system. Then, $V(x_{t+1}) \leq \gamma^2 V(x_t)$ for all $t \geq 0$. It follows that $V(x_t) \leq \gamma^{2t} V(x_0)$. So we have

$$\|x_t\|_2^2 = \frac{x_t^T x_t}{x_t^T P x_t} x_t^T P x_t \leq \sup_{x \neq 0} \frac{x^T x}{x^T P x} V(x_t) \leq \lambda_{\min}(P)^{-1} \gamma^{2t} V(x_0),$$

and thus,

$$\|x_t\|_2^{1/t} \leq \lambda_{\min}(P)^{-1/2t} \gamma V(x_0)^{1/2t}.$$

Taking the limit as $t \to \infty$ we get $\kappa \leq \gamma$.

(b) The given condition on V is equivalent to the matrix inequalities

$$A^{(i)T} P A^{(i)} \preceq \gamma^2 P, \quad i = 1, \ldots, K.$$

For any fixed γ, these are linear matrix inequalities in P, hence convex constraints. Note that the LMIs above are homogeneous in P. Therefore, without loss of generality, we can require $P \succeq I$ in order to enforce P to be positive definite. Thus, there is a quadratic Lyapunov function that establishes the bound γ if and only if the LMIs

$$A^{(i)T} P A^{(i)} \preceq \gamma^2 P, \quad i = 1, \ldots, K, \qquad P \succeq I$$

are feasible. This defines a convex set of P, so finding the smallest possible value of γ is a quasiconvex problem. Then, we can use bisection on γ to solve this problem.

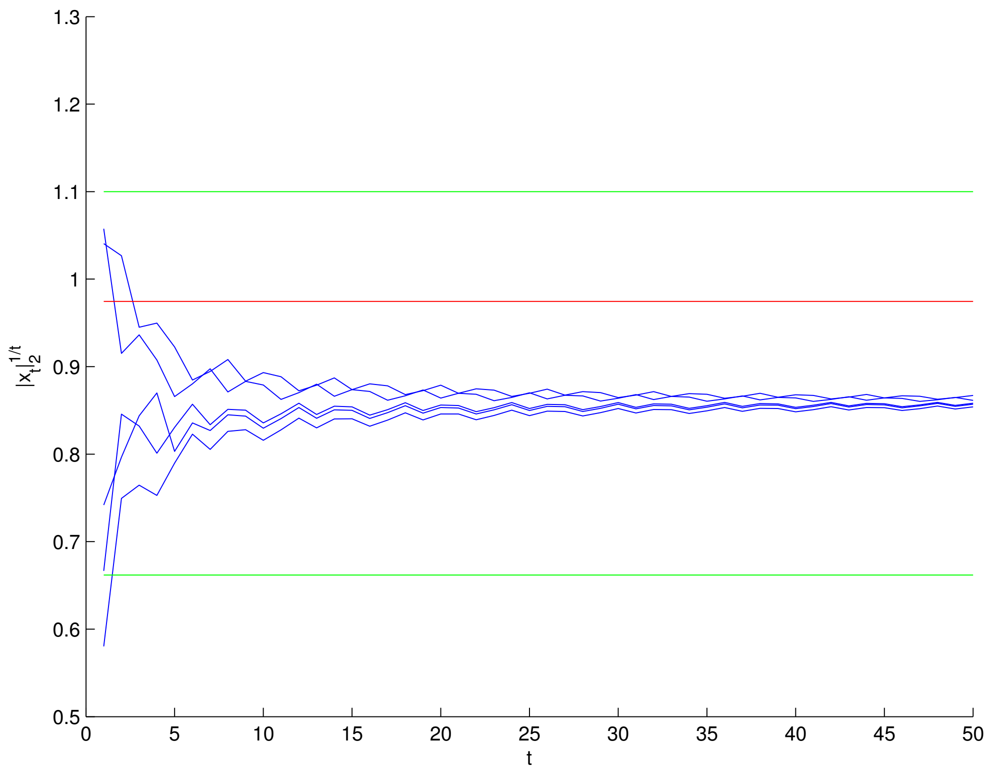
(c) We find that the best bound obtained by the method above is $\gamma = 0.97$. The following code solves the problem.

```
% lyapunov analysis of a dynamical system
lyap_exp_bound_data;

% compute lower and upper bounds for bisection
l = 0; u = 0;
for i = 1:K
    l = max(l, max(abs(eig(A{i}))));
    u = max(u, norm(A{i}));
end
bisection_tol = 1e-4;

while u-l >= bisection_tol
    fprintf('bisection bounds: %f %f\n', l, u);
    gamma = (l+u)/2;
    cvx_begin quiet
```

```
        variable P(n, n) symmetric
        minimize trace(P)
        subject to
            for i = 1:K
                gamma^2*P - A{i}'*P*A{i} == semidefinite(n)
            end
            P-eye(n) == semidefinite(n)
    cvx_end

    if strcmp(cvx_status, 'Solved')
        u = gamma;
        bound = gamma;
        P_opt = P;
    else
        l = gamma;
    end
end

fprintf('smallest value of gamma = %f\n', bound);
```

We mention the initial lower and upper bound used in the code, that we didn't ask you to explore. Suppose that the initial state was an eigenvector of some $A^{(i)}$ with eigenvalue λ, and that this particular $A^{(i)}$ was chosen at every time step. Under this condition, it is easy to see that $|\lambda|$ gives a lower bound on γ. On the other hand, the ratio $\|x_{t+1}\|_2/\|x_t\|_2$ is bounded by $\max_i \|A^{(i)}\|$, so we obtain an upper bound on γ.

(d) The figure shows five random trajectories in blue, and the γ bound in red. The initial lower and upper bound used in the bisection method are shown in green. The trajectories suggest that the Lyapunov exponent of the system is 0.86 or higher.



The following code generates the plot.

```
% approximate worst-case trajectory
% simulation of a dynamical system
clf; hold on;
T = 50;
for traj = 1:5
    x = randn(n, 1); % random initial state
    x = 1.1*x/norm(x);
    ys = [];
    for t = 1:T
        best = 0;
        for i = 1:K
            if best < x'*A{i}'*P_opt*A{i}*x
                best = x'*A{i}'*P_opt*A{i}*x;
                x_next = A{i}*x;
            end
        end
        x = x_next;
        ys(end+1) = norm(x)^(1/t);
    end
    plot(1:T, ys);
```

```
end

l = 0; u = 0;
for i = 1:K
    l = max(l, max(abs(eig(A{i}))));
    u = max(u, norm(A{i}));
end

plot([1 T], [bound bound], 'r', ...
     [1 T], [l l], 'g', ...
     [1 T], [u u], 'g');

xlabel('t'); ylabel('|x_t|_2^{1/t}');
hold off;

print -depsc lyap_exp_bound.eps;
```