EE364a: Convex Optimization I S. Boyd
March 15–16 or March 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 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; 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 937. 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. *Minimum time speed profile along a road.* A vehicle of mass $m > 0$ moves along a road in $\mathbf{R}^3$, which is piecewise linear with given knot points $p_1, \ldots, p_{N+1} \in \mathbf{R}^3$, starting at p_1 and ending at p_{N+1}. We let $h_i = (p_i)_3$, the z-coordinate of the knot point; these are the heights of the knot points (above sea-level, say). For your convenience, these knot points are equidistant, *i.e.*, $\|p_{i+1} - p_i\|_2 = d$ for all i. (The points give an arc-length parametrization of the road.) We let $s_i > 0$ denote the (constant) vehicle speed as it moves along road segment i, from p_i to p_{i+1}, for $i = 1, \ldots, N$, and $s_{N+1} \geq 0$ denote the vehicle speed after it passes through knot point p_{N+1}. Our goal is to minimize the total time to traverse the road, which we denote T.

   We let $f_i \geq 0$ denote the total fuel burnt while traversing the ith segment. This fuel burn is turned into an increase in vehicle energy given by ηf_i, where $\eta > 0$ is a constant that includes the engine efficiency and the energy content of the fuel. While traversing the ith road segment the vehicle is subject to a drag force, given by $C_D s_i^2$, where $C_D > 0$ is the coefficient of drag, which results in an energy loss $dC_D s_i^2$.

   We derive equations that relate these quantities via energy balance:

$$\frac{1}{2}ms_{i+1}^2 + mgh_{i+1} = \frac{1}{2}ms_i^2 + mgh_i + \eta f_i - dC_D s_i^2, \quad i = 1, \ldots, N,$$

   where $g = 9.8$ is the gravitational acceleration. The lefthand side is the total vehicle energy (kinetic plus potential) after it passes through knot point p_{i+1}; the righthand side is the total vehicle energy after it passes through knot point p_i, plus the energy gain from the fuel burn, minus the energy lost to drag. To set up the first vehicle speed s_1 requires an additional initial fuel burn f_0, with $\eta f_0 = \frac{1}{2}ms_1^2$.

   Fuel is also used to power the on-board system of the vehicle. The total fuel used for this purpose is f_{ob}, where $\eta f_{\text{ob}} = TP$, where $P > 0$ is the (constant) power consumption of the on-board system. We have a fuel capacity constraint: $\sum_{i=0}^{N} f_i + f_{\text{ob}} \leq F$, where $F > 0$ is the total initial fuel.

   The problem data are m, d, $h_1, \ldots, h_{N+1}$, η, C_D, P, and F. (You don't need the knot points p_i.)

   (a) Explain how to find the fuel burn levels $f_0, \ldots, f_N$ that minimize the time T, subject to the constraints.

   (b) Carry out the method described in part (a) for the problem instance with data given in `min_time_speed_data.m`. Give the optimal time $T^\star$, and compare it to the time T^{unif} achieved if the fuel for propulsion were burned uniformly, *i.e.*, $f_0 = \cdots = f_N$. For each of these cases, plot speed versus distance along the road, using the plotting code in the data file as a template.

   Solution.

(a) The time to traverse the ith segment is d/s_i, so the total time is

$$T = \sum_{i=1}^{N} \frac{d}{s_i}.$$

We are to solve the problem

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} d/s_i \\ \text{subject to} & \frac{1}{2}ms_{i+1}^2 + mgh_{i+1} = \frac{1}{2}ms_i^2 + mgh_i + \eta f_i - dC_D s_i^2, \quad i = 1, \ldots, N \\ & \sum_{i=0}^{N} f_i + PT/\eta \leq F \\ & f_i \geq 0, \quad i = 0, \ldots, N \\ & \eta f_0 = \frac{1}{2}ms_1^2, \end{array}$$

with variables f_i and s_i. The domain of the objective gives the implicit constraint $s_i > 0$, $i = 1, \ldots, N$. The objective is convex, but unfortunately, the equality constraints are not linear because of the s_i^2 terms. So we will try introducing the new variables $z_i = s_i^2$. Since $s_i \geq 0$, we can recover speed from the new variables using $s_i = \sqrt{z_i}$. Note that z_i is proportional to the kinetic energy; therefore this change of variables can be thought of as solving the problem in terms of kinetic energy instead of speed. Our equality constraints now become

$$\frac{1}{2}mz_{i+1} + mgh_{i+1} = \frac{1}{2}mz_i + mgh_i + \eta f_i - dC_D z_i, \quad i = 1, \ldots, N$$

and $\eta f_0 = \frac{1}{2}mz_1$, which are linear equations in the variables z_i and f_i. We now return to our objective which, under this change of variables, becomes

$$T = \sum_{i=1}^{N} \frac{d}{s_i} = \sum_{i=1}^{N} dz_i^{-1/2},$$

which is convex since $z_i^{-1/2}$ is convex. Having shown that T is convex in our new variables, it is clear that the fuel capacity constraint

$$\sum_{i=0}^{N} f_i + PT/\eta = \sum_{i=0}^{N} f_i + P/\eta \sum_{i=1}^{N} dz_i^{-1/2} \leq F$$

is a convex constraint. We can therefore find $f_0, \ldots, f_N$ by solving the convex optimization problem

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} dz_i^{-1/2} \\ \text{subject to} & \frac{1}{2}mz_{i+1} + mgh_{i+1} = \frac{1}{2}mz_i + mgh_i + \eta f_i - dC_D z_i, \quad i = 1, \ldots, N \\ & \sum_{i=0}^{N} f_i + P/\eta \sum_{i=1}^{N} d(z_i)^{-1/2} \leq F \\ & f_i \geq 0, \quad i = 0, \ldots, N \\ & \eta f_0 = \frac{1}{2}mz_1, \end{array}$$

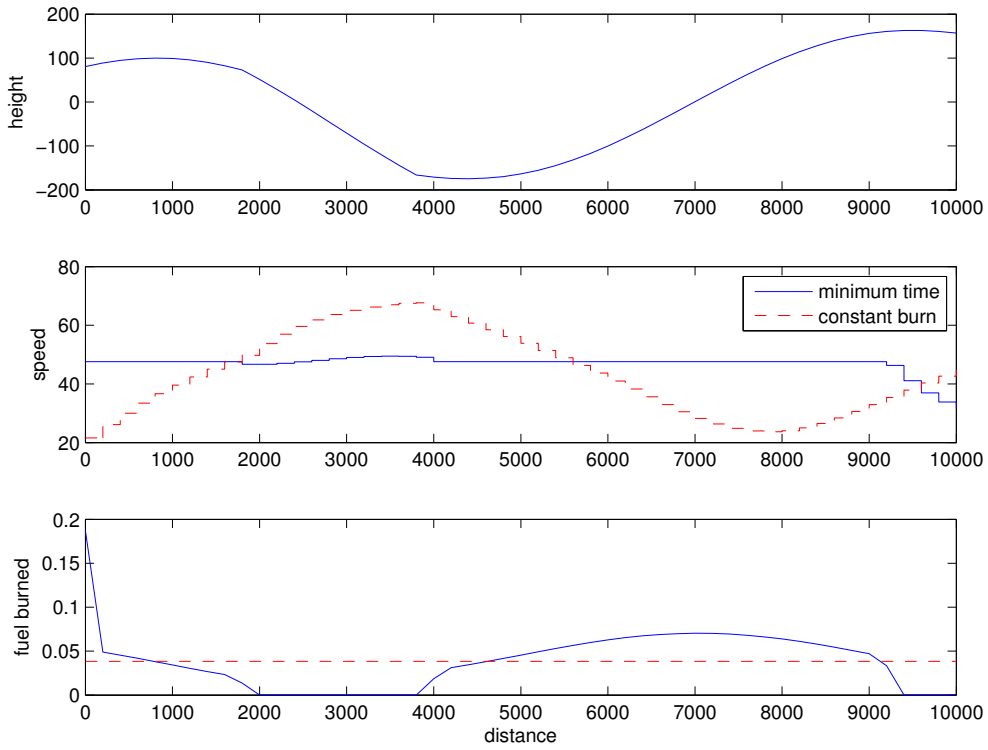
with variables $f_0, \ldots, f_N$ and $z_1, \ldots, z_N$. Then we recover s_i using $s_i = \sqrt{z_i}$.

(b) The following code solves the problem:

```
% Minimum time speed profile along a road.
min_time_speed_data;

% minimum time
cvx_begin
variables z(N+1) f(N+1)
minimize sum(d*inv_pos(sqrt(z(1:N))))
subject to
.5*m*z(2:N+1)+m*g*h(2:N+1) ==...
    .5*m*z(1:N)+m*g*h(1:N)+eta*f(2:N+1)-d*C_D*z(1:N)
sum(f)+P/eta*sum(d*inv_pos(sqrt(z(1:N)))) <= F
f >= 0
eta*f(1) == .5*m*z(1)
cvx_end

T = cvx_optval

% constant fuel burn
cvx_solver sdpt3
% sedumi fails, but only on this 1 part of 1 problem
% from the entire exam, and only if you replaced
% the vector f, by the variable fc as below
cvx_begin
variables zc(N+1) fc
minimize sum(d*inv_pos(sqrt(zc(1:N))))
subject to
.5*m*zc(2:N+1)+m*g*h(2:N+1) ==...
    .5*m*zc(1:N)+m*g*h(1:N)+eta*fc-d*C_D*zc(1:N)
(N+1)*fc+P/eta*sum(d*inv_pos(sqrt(zc(1:N)))) <= F
fc >= 0
eta*fc == .5*m*zc(1)
cvx_end

T_unif = cvx_optval

figure
subplot(3,1,1)
plot((0:N)*d,h);
ylabel('height');
subplot(3,1,2)
stairs((0:N)*d,sqrt(z),'b');
```

```
hold on
stairs((0:N)*d,sqrt(zc),'--r');
ylabel('speed')
legend('minimum time','constant burn')
subplot(3,1,3)
plot((0:N)*d,f,'b');
hold on
plot((0:N)*d,fc*ones(N+1,1),'--r')
xlabel('distance')
ylabel('fuel burned')

print -depsc min_time_speed;
```

We get $T^* = 213.26$ and $T^{\text{unif}} = 258.48$. Note that you can find T^{unif} by replacing the f_i's by a single parameter f_c as we did, or by constraining all of the f_i's to be equal. The plot below shows the speed and fuel burn profiles for the two cases.



We see that the optimal fuel burn attempts to keep a constant velocity, except near the end of the trajectory, where it coasts to the finish line. The uniform fuel burn wastes fuel (and therefore loses time) by burning fuel on the downhill parts, leading to a large speed, and therefore, large loss due to drag.

2. *Polynomial approximation of inverse using eigenvalue information.* We seek a polynomial of degree k, $p(a) = c_0 + c_1 a + c_2 a^2 + \cdots + c_k a^k$, for which

$$p(A) = c_0 I + c_1 A + c_2 A^2 \cdots + c_k A^k$$

is an approximate inverse of the nonsingular matrix A, for all $A \in \mathcal{A} \subset \mathbf{R}^{n\times n}$. When $\hat{x} = p(A)b$ is used as an approximate solution of the linear equation $Ax = b$, the associated residual norm is $\|A(p(A)b) - b\|_2$. We will judge our polynomial (*i.e.*, the coefficients $c_0, \ldots, c_k$) by the worst case residual over $A \in \mathcal{A}$ and b in the unit ball:

$$R^{\mathrm{wc}} = \sup_{A\in\mathcal{A},\ \|b\|_2\leq 1} \|A(p(A)b) - b\|_2.$$

The set of matrices we take is $\mathcal{A} = \{A \in \mathbf{S}^n \mid \sigma(A) \subseteq \Omega\}$, where $\sigma(A)$ is the set of eigenvalues of A (*i.e.*, its spectrum), and $\Omega \subset \mathbf{R}$ is a union of a set of intervals (that do not contain 0).

(a) Explain how to find coefficients $c_0^\star, \ldots, c_k^\star$ that minimize R^{wc}. Your solution can involve expressions that involve the supremum of a polynomial (with scalar argument) over an interval.

(b) Carry out your method for $k = 4$ and $\Omega = [-0.6, -0.3] \cup [0.7, 1.8]$. You can replace the supremum of a polynomial over Ω by a maximum over uniformly spaced (within each interval) points in Ω, with spacing 0.01. Give the optimal value $R^{\mathrm{wc}\star}$ and the optimal coefficients $c^\star = (c_0^\star, \ldots, c_k^\star)$.

Remarks. (Not needed to solve the problem.)

- The approximate inverse $p(A)b$ would be computed by recursively, requiring the multiplication of A with a vector k times.
- This approximate inverse could be used as a preconditioner for an iterative method.
- The Cayley-Hamilton theorem tells us that the inverse of any (invertible) matrix is a polynomial of degree $n-1$ of the matrix. Our hope here, however, is to get a single polynomial, of relatively low degree, that serves as an approximate inverse for many different matrices.

Solution.

(a) We can rewrite

$$R^{\mathrm{wc}} = \sup_{A\in\mathcal{A}} \sup_{\|b\|_2\leq 1} \|A(p(A)b) - b\|_2,$$

and recognize the inner supremum as the definition of the spectral norm of $Ap(A) - I$. If A is symmetric, then $Ap(A) - I$ is also symmetric, and its spectral norm is the largest absolute value of its eigenvalues.

Let QDQ^T be an eigenvalue decomposition of A, with Q orthogonal and D diagonal. Then

$$\begin{aligned} Ap(A) - I &= c_0 A + c_1 A^2 + \cdots + c_k A^{k+1} - I \\ &= c_0 QDQ^T + c_1 QD^2Q^T + \cdots + c_k QD^{k+1}Q^T - I \\ &= Q(c_0 D + c_1 D^2 + \cdots + c_k D^{k+1} - I)Q^T \\ &= Q(Dp(D) - I)Q^T, \end{aligned}$$

which shows that $\lambda p(\lambda) - 1 \in \sigma(Ap(A) - I)$ if $\lambda \in \sigma(A)$.
We can then rewrite

$$\begin{aligned} R^{\mathrm{wc}} &= \sup_{A \in \mathcal{A}} \|Ap(A) - I\|_2 \\ &= \sup_{A \in \mathcal{A}} \sup_{\lambda \in \sigma(A)} |\lambda p(\lambda) - 1| \\ &= \sup_{\lambda \in \Omega} |\lambda p(\lambda) - 1| \\ &= \sup_{\lambda \in \Omega} |c_0 \lambda + c_1 \lambda^2 + \cdots + c_k \lambda^{k+1} - 1|, \end{aligned}$$

and note that R^{wc} is a convex function of c since, for any λ, $c_0\lambda + c_1\lambda^2 + \cdots + c_k\lambda^{k+1} - 1$ is an affine function of c. We can write the optimization problem as

$$\text{minimize} \quad \sup_{\lambda \in \Omega} |c_0 \lambda + c_1 \lambda^2 + \cdots + c_k \lambda^{k+1} - 1|.$$

This problem can be converted exactly into an SDP (since the supremum of a polynomial has an LMI representation), but for any practical purpose simple sampling of Ω is fine.

(b) Let $\lambda_1, \ldots, \lambda_N$ be uniformly spaced (within each interval contained in Ω) sample points, with spacing 0.01. We approximate the objective by

$$\max_{i=1,\ldots,N} |c_0 \lambda_i + c_1 \lambda_i^2 + \cdots + c_k \lambda_i^{k+1} - 1|.$$

Define $\Lambda \in \mathbf{R}^{N \times (k+1)}$ as $\Lambda_{ij} = \lambda_i^j$. This allows us to write the (approximate) optimization problem as

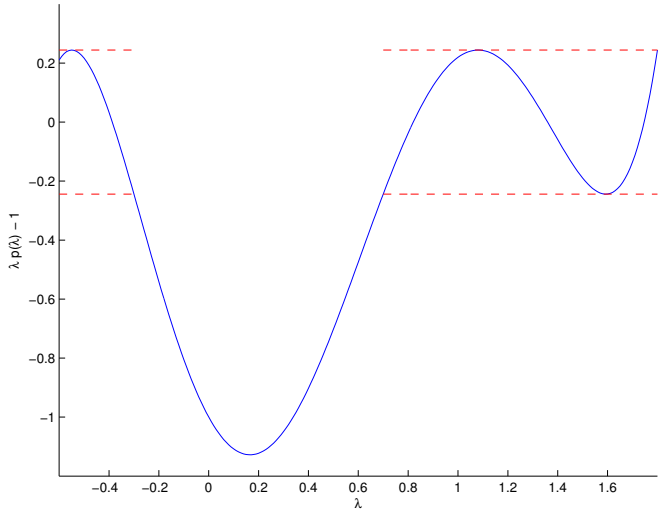
$$\text{minimize} \quad \|\Lambda c - \mathbf{1}\|_\infty.$$

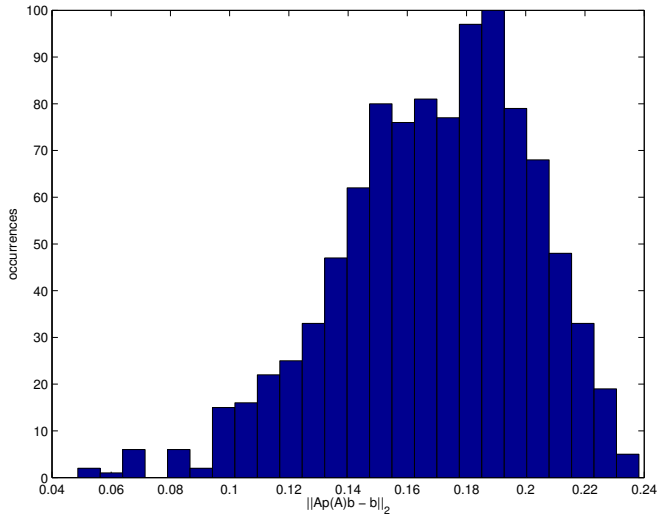
We find that $R^{\mathrm{wc}\star} = 0.2441$, with coefficients

$$c^\star = (-1.54, 4.43, 1.98, -5.61, 1.96).$$

That's impressive: A single polynomial of degree 4 serves as a crude inverse for a whole family of matrices! We plot $\lambda p(\lambda) - 1$ with a blue line over the interval $[-0.6, 1.8]$, and indicate Ω and the bounds $\pm R^{\mathrm{wc}\star}$ with a red dashed line.

We didn't ask you to do this, but we also compute $\|Ap(A)b-b\|_2$ for 1000 random instances of $A \in \mathcal{A}$ with $n = 10$, and b with $\|b\|_2 = 1$. We generate A by sampling eigenvalues uniformly from Ω, placing them in a diagonal matrix D, and forming $A = QDQ^T$, for a random orthogonal matrix Q (which we generate by taking the QR decomposition of a matrix with Gaussian entries). We generate b from a Gaussian distribution and normalize (which gives a uniform distribution on the unit sphere). Of course, the distributions don't matter. A histogram and the code are given below. Note that the worst case error across the samples is a bit under 0.24, quite consistent with our value of $R^{\mathrm{wc}\star}$.






```
% Polynomial approximation of inverse using eigenvalue information.
k = 4; % k degree polynomials
l1 = -.6;
u1 = -.3;
l2 = .7;
```

```
u2 = 1.8;
eta = 0.01;

lambdas = [l1:eta:u1 l2:eta:u2]';

Lambda = lambdas;
for i = 1:k
    Lambda(:,end+1) = Lambda(:,end).*lambdas;
end

cvx_begin
    variable c(k+1)
    minimize( norm(Lambda*c - 1,inf) )
cvx_end
R_wc = cvx_optval;

lambdas_full = [l1:eta:u2]';
Lambda_full = lambdas_full;
for i = 1:k
    Lambda_full(:,end+1) = Lambda_full(:,end).*lambdas_full;
end

y2 = Lambda_full*c - 1;
figure
hold on
plot(lambdas_full,y2)
plot(l1:eta:u1,0*(l1:eta:u1) + R_wc,'--r',...
     l1:eta:u1,0*(l1:eta:u1) - R_wc,'--r',...
     l2:eta:u2,0*(l2:eta:u2) + R_wc,'--r',...
     l2:eta:u2,0*(l2:eta:u2) - R_wc,'--r')

ylabel('\lambda p(\lambda) - 1')
xlabel('\lambda')
axis([l1 u2 -1.2 .4])
print '-depsc' 'poly_approx_inv.eps'


%testing code
randn('state',0)
rand('state',0)
n = 10;
runs = 1000;
```

```
r = (u1-l1)/(u2-l2 + u1-l1);

errors = zeros(runs,1);
for i = 1:runs
    %create random eigenvalues in [l1,u1] \cup [l2,u2]
    a = rand(n,1);
    a = l1 + (l2-u1)*(sign(a-r)+1)/2 + (u2-l2 + u1-l1)*a;
    [Q R] = qr(rand(n,n));
    A = Q*diag(a)*Q';

    b = randn(n,1);
    b = b/norm(b);

    q = -b;
    for j = 1:k+1
        q = q + c(j)*A^j*b;
    end
    errors(i) = norm(q);
end

figure
hist(errors,25)
xlabel('||Ap(A)b - b||_2')
ylabel('occurrences')
print -depsc poly_approx_inv_hist.eps
```

3. *Fitting a generalized additive regression model.* A *generalized additive model* has the form

$$f(x) = \alpha + \sum_{j=1}^{n} f_j(x_j),$$

for $x \in \mathbf{R}^n$, where $\alpha \in \mathbf{R}$ is the offset, and $f_j : \mathbf{R} \to \mathbf{R}$, with $f_j(0) = 0$. The functions f_j are called the *regressor functions*. When each f_j is linear, *i.e.*, has the form $w_j x_j$, the generalized additive model is the same as the standard (linear) regression model. Roughly speaking, a generalized additive model takes into account nonlinearities in each regressor x_j, but not nonlinear interactions among the regressors. To visualize a generalized additive model, it is common to plot each regressor function (when n is not too large).

We will restrict the functions f_j to be piecewise-affine, with given knot points $p_1 < \cdots < p_K$. This means that f_j is affine on the intervals $(-\infty, p_1]$, $[p_1, p_2]$, $\ldots$, $[p_{K-1}, p_K]$, $[p_K, \infty)$, and continuous at $p_1, \ldots, p_K$. Let C denote the total (absolute value of) change in slope across all regressor functions and all knot points. The value C is a measure of nonlinearity of the regressor functions; when $C = 0$, the generalized additive model reduces to a linear regression model.

Now suppose we observe samples or data $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)}) \in \mathbf{R}^n \times \mathbf{R}$, and wish to fit a generalized additive model to the data. We choose the offset and the regressor functions to minimize

$$\frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - f(x^{(i)})^2 + \lambda C,$$

where $\lambda > 0$ is a regularization parameter. (The first term is the mean-square error.)

(a) Explain how to solve this problem using convex optimization.

(b) Carry out the method of part (a) using the data in the file `gen_add_reg_data.m`. This file contains the data, given as an $N \times n$ matrix `X` (whose rows are $(x^{(i)})^T$), a column vector `y` (which give $y^{(i)}$), a vector `p` that gives the knot points, and the scalar `lambda`.

Give the mean-square error achieved by your generalized additive regression model. Compare the estimated and true regressor functions in a 3×3 array of plots (using the plotting code in the data file as a template), over the range $-10 \leq x_i \leq 10$. The true regressor functions (to be used only for plotting, of course) are given in the cell array `f`.

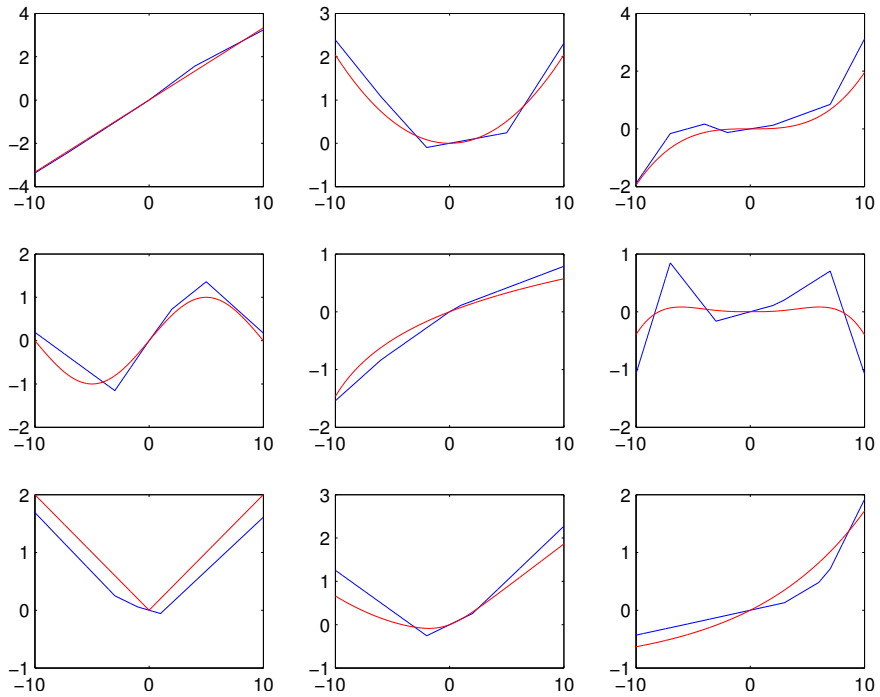
Hints.

- You can represent each regressor function f_j as a linear combination of the basis functions $b_0(u) = u$ and $b_i(u) = (u - p_k)_+ - (-p_k)_+$ for $k = 1, 2, \ldots, K$, where $(a)_+ = \max\{a, 0\}$.

- You might find the matrix $\texttt{XX} = \begin{bmatrix} b_0(\texttt{X}) & b_1(\texttt{X}) & \cdots & b_K(\texttt{X}) \end{bmatrix}$ useful.

Solution. There is not much more to say beyond showing the code and the plot.

```
% Fitting a generalized additive regression model.
gen_add_reg_data;

%build an augmented data matrix XX
XX=X;
for ii=1:K
    XX=[XX,max(0,X-p(ii))+min(p(ii),0)];
end

%Perform regression
cvx_begin
    variables alpha c(9*(K+1))
    minimize(1/N*sum_square(y-alpha-XX*c)+lambda*norm(c,1))
cvx_end

%Plot functions.
xx=linspace(-10,10,1024);
yy=zeros(9,1024);
figure
for jj=1:9
    yy(jj,:)=c(jj)*xx;
    for ii=1:K
        yy(jj,:)=yy(jj,:)+c(ii*9+jj)*(pos(xx-p(ii))-pos(-p(ii)));
    end
    subplot(3,3,jj);
    plot(xx,yy(jj,:));
    hold on;
    plot(xx,f{jj}(xx),'r')
end


print -depsc gen_add_reg.eps
```



The blue and red lines correspond to the estimated and true regressors, respectively.

4. *Maximum likelihood estimation for an affinely transformed distribution.* Let z be a random variable on $\mathbf{R}^n$ with density $p_z(u) = \exp -\phi(\|u\|_2)$, where $\phi : \mathbf{R} \to \mathbf{R}$ is convex and increasing. Examples of such distributions include the standard normal $\mathcal{N}(0, \sigma^2 I)$, with $\phi(u) = (u)_+^2 + \alpha$, and the multivariable Laplacian distribution, with $\phi(u) = (u)_+ + \beta$, where α and β are normalizing constants, and $(a)_+ = \max\{a, 0\}$. Now let x be the random variable $x = Az + b$, where $A \in \mathbf{R}^{n\times n}$ is nonsingular. The distribution of x is parametrized by A and b.

   Suppose $x_1, \ldots, x_N$ are independent samples from the distribution of x. Explain how to find a maximum likelihood estimate of A and b using convex optimization. If you make any further assumptions about A and b (beyond invertiblility of A), you must justify it.

   Hint. The density of $x = Az + b$ is given by

$$p_x(v) = \frac{1}{|\det A|} p_z(A^{-1}(v - b)).$$

   Solution. The density of $x = Az + b$ is given by

$$p_x(v) = \frac{1}{|\det A|} \exp -\phi(\|A^{-1}(v - b)\|_2).$$

   We first observe that the density with parameters (A, b) is the same as the density with parameters (AQ, b), for any orthogonal matrix Q, since

$$|\det(AQ)| = |\det A||\det Q| = |\det A|,$$

   and

$$\|(AQ)^{-1}(v - b)\|_2 = \|Q^T A^{-1}(v - b)\|_2 = \|A^{-1}(v - b)\|_2.$$

   Let A have SVD $A = U\Sigma V^T$. Choosing $Q = VU^T$, we see that $AQ = U\Sigma U^T \in \mathbf{S}^n_{++}$. So we can always assume that $A \in \mathbf{S}^n_{++}$.

   The log-likelihood function for a single sample x is

$$\ell(A, b) = -\log\det A - \phi(\|A^{-1}(x - b)\|_2),$$

   so for N independent samples, we have log-likelihood function

$$\ell(A, b) = -N \log\det A - \sum_{i=1}^{N} \phi(\|A^{-1}(x_i - b)\|_2).$$

   We must maximize ℓ over $A \in \mathbf{S}^n_{++}$ and $b \in \mathbf{R}^n$. It's not concave in these parameters, but we will use instead the parameters

$$B = A^{-1} \in \mathbf{S}^n_{++}, \qquad c = A^{-1}b \in \mathbf{R}^n.$$

From B and c we can recover A and b as

$$A = B^{-1}, \qquad b = B^{-1}c.$$

In terms of B and c, the log-likelihood function is

$$\tilde{\ell}(B, c) = N \log\det B - \sum_{i=1}^{N} \phi(\|Bx_i - c\|_2),$$

which is concave. To see this, we note that the first term is concave in B, and the second term is concave since $\|Bx_i - c\|_2$ is convex in (B, c), and by the composition rule, $\phi(\|Bx_i - c\|_2)$ is convex.

So we just maximize $\tilde{\ell}(B, c)$ over $B \in \mathbf{S}^n_{++}$, $c \in \mathbf{R}^n$, and then get A and b as described above.

5. *Functions of a random variable with log-concave density.* Suppose the random variable X on $\mathbf{R}^n$ has log-concave density, and let $Y = g(X)$, where $g : \mathbf{R}^n \to \mathbf{R}$. For each of the following statements, either give a counterexample, or show that the statement is true.

   (a) If g is affine and not constant, then Y has log-concave density.

   (b) If g is convex, then $\mathbf{Prob}(Y \leq a)$ is a log-concave function of a.

   (c) If g is concave, then $\mathbf{E}\,((Y-a)_+)$ is a convex and log-concave function of a. (This quantity is called the tail expectation of Y; you can assume it exists. We define $(s)_+$ as $(s)_+ = \max\{s, 0\}$.)

   Solution.

   (a) This one is **true**. Let p be the density of X, and let $g(x) = c^T x + d$, with $c \neq 0$ (otherwise g would be constant). Since g is not constant, we conclude that Y has a density p_Y.

   With $\delta a > 0$, define the function

$$h(x,a) = \begin{cases} 1 & a \leq g(x) \leq a + \delta a \\ 0 & \text{otherwise,} \end{cases}$$

   which is the $0-1$ indicator function of the convex set $\{(x,a) \mid a \leq g(x) \leq a+\delta a\}$. The $0-1$ indicator function of a convex set is log-concave, so by the integration rule it follows that

$$\int p(x)h(x,a)\,dx = \mathbf{E}\,h(X,a) = \mathbf{Prob}(a \leq Y \leq a + \delta a)$$

   is log-concave in a. It follows that

$$\frac{\mathbf{Prob}(a \leq Y \leq a + \delta a)}{\delta a}$$

   is log-concave (since $\delta a > 0$). Taking $\delta a \to 0$, this converges to $p_Y(a)$, which we conclude is log-concave.

   (b) This one is **true**. Here we define the function

$$h(x,a) = \begin{cases} 1 & g(x) \leq a \\ 0 & \text{otherwise,} \end{cases}$$

   which is the $0-1$ indicator function of the convex set $\mathbf{epi}\, g = \{(x,a) \mid g(x) \leq a\}$, and therefore log-concave. By the integration rule we get that

$$\int p(x)h(x,a)\,dx = \mathbf{E}\,h(X,a) = \mathbf{Prob}(Y \leq a)$$

   is log-concave in a.

   If we assume that g is concave, and we switch the inequality, we conclude that $\mathbf{Prob}(Y \geq a)$ is log-concave in a. (We'll use this below.)

(c) This one is **true**. Convexity of the tail expectation holds for *any* random variable, so it has has nothing to do with g, and it has nothing to do with log-concavity of the density of X. For *any* random variable Y on $\mathbf{R}$, we have

$$\frac{d}{da}\,\mathbf{E}(Y-a)_+ = -\,\mathbf{Prob}(Y \geq a).$$

The righthand side is nondecreasing in a, so the tail expectation has nondecreasing derivative, which means it is a convex function.

Now let's show that the tail expectation is log-concave. One simple method is to use the formula above to note that

$$\mathbf{E}(Y-a)_+ = \int_a^\infty \mathbf{Prob}(Y \geq b)\; db.$$

The integration rule for log-concave functions tells us that this is log-concave.

We can also give a direct proof following the style of the ones given above. We define g as $h(x,a) = (g(x)-a)_+$. This function is log-concave. First, its domain is $\{(x,a) \mid g(x) > a\}$, which is convex. Concavity of $\log h(x,a) = \log(g(x)-a)$ follows from the composition rule: log is concave and increasing, and $g(x)-a$ is concave in (x,a). So by the integration rule we get

$$\int p(x)h(x,a)\; dx = \mathbf{E}(g(x)-a)_+$$

is log-concave in a.

6. *Affine policy.* We consider a family of LPs, parametrized by the random variable u, which is uniformly distributed on $\mathcal{U} = [-1, 1]^p$,

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \preceq b(u), \end{array}$$

where $x \in \mathbf{R}^n$, $A \in \mathbf{R}^{m\times n}$, and $b(u) = b_0 + Bu \in \mathbf{R}^m$ is an affine function of u. You can think of u_i as representing a deviation of the ith parameter from its nominal value. The parameters might represent (deviations in) levels of resources available, or other varying limits.

The problem is to be solved many times; in each time, the value of u (*i.e.*, a sample) is given, and then the decision variable x is chosen. The mapping from u into the decision variable $x(u)$ is called the *policy*, since it gives the decision variable value for each value of u. When enough time and computing hardware is available, we can simply solve the LP for each new value of u; this is an optimal policy, which we denote $x^\star(u)$.

In some applications, however, the decision $x(u)$ must be made very quickly, so solving the LP is not an option. Instead we seek a suboptimal policy, which is affine: $x^{\text{aff}}(u) = x_0 + Ku$, where x_0 is called the *nominal decision* and $K \in \mathbf{R}^{n\times p}$ is called the *feedback gain matrix*. (Roughly speaking, x_0 is our guess of x before the value of u has been revealed; Ku is our modification of this guess, once we know u.) We determine the policy (*i.e.*, suitable values for x_0 and K) ahead of time; we can then evaluate the policy (that is, find $x^{\text{aff}}(u)$ given u) very quickly, by matrix multiplication and addition.

We will choose x_0 and K in order to minimize the expected value of the objective, while insisting that for any value of u, feasibility is maintained:

$$\begin{array}{ll} \text{minimize} & \mathbf{E}\, c^T x^{\text{aff}}(u) \\ \text{subject to} & Ax^{\text{aff}}(u) \preceq b(u) \quad \forall u \in \mathcal{U}. \end{array}$$

The variables here are x_0 and K. The expectation in the objective is over u, and the constraint requires that $Ax^{\text{aff}}(u) \preceq b(u)$ hold almost surely.

(a) Explain how to find optimal values of x_0 and K by solving a standard explicit convex optimization problem (*i.e.*, one that does not involve an expectation or an infinite number of constraints, as the one above does.) The numbers of variables or constraints in your formulation should not grow exponentially with the problem dimensions n, p, or m.

(b) Carry out your method on the data given in `affine_pol_data.m`. To evaluate your affine policy, generate 100 independent samples of u, and for each value, compute the objective value of the affine policy, $c^T x^{\text{aff}}(u)$, and of the optimal policy, $c^T x^\star(u)$. Scatter plot the objective value of the affine policy (y-axis) versus the objective value of the optimal policy (x-axis), and include the line $y = x$ on the plot. Report the average values of $c^T x^{\text{aff}}(u)$ and $c^T x^\star(u)$ over your samples. (These are estimates of $\mathbf{E}\, c^T x^{\text{aff}}(u)$ and $\mathbf{E}\, c^T x^\star(u)$. The first number, by the way, can be found exactly.)

Solution. Let's start with the objective. We compute the expected value of the affine policy as

$$\mathbf{E}\, c^T(x_0 + Ku) = c^T x_0 + (K^T c)^T\, \mathbf{E}\, u = c^T x_0.$$

Now let's look at the constraints, which we write out in terms of its entries:

$$(Ax_0)_i + \sup_{u \in \mathcal{U}}((AK - B)u)_i \leq (b_0)_i, \quad i = 1, \ldots, m.$$

This turns into the explicit constraint

$$(Ax_0)_i + \|(AK - B)_i\|_1 \leq (b_0)_i, \quad i = 1, \ldots, m.$$

Here $(AK - B)_i$ is the ith row of the matrix $A - BK$.

So we can find an optimal affine policy by solving the problem (which can be transformed to an LP)

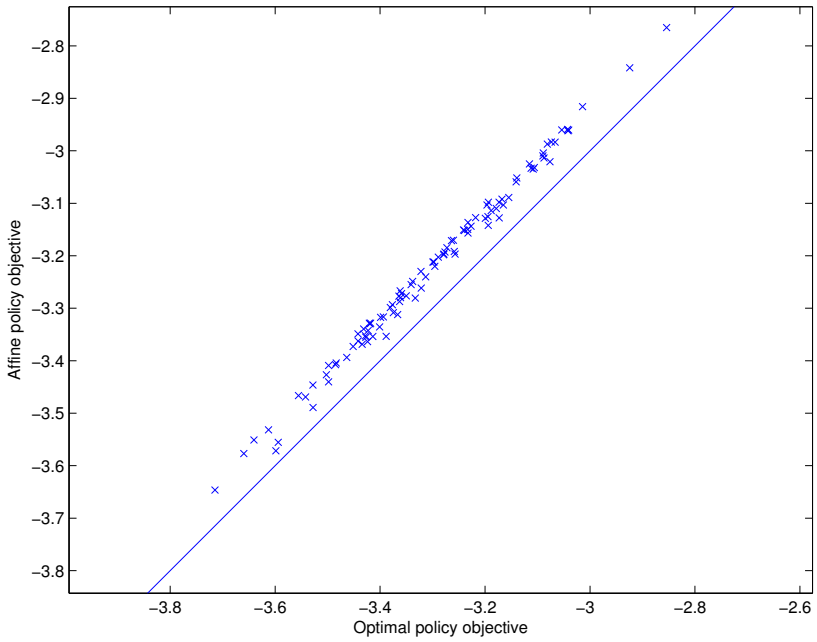
$$\begin{array}{ll} \text{minimize} & c^T x_0 \\ \text{subject to} & (Ax_0)_i + \|(AK - B)_i\|_1 \leq (b_0)_i, \quad i = 1, \ldots, m, \end{array}$$

with variables x_0 and K.

The code below implements this.

```
% Affine policy.
affine_pol_data;

% compute affine policy
cvx_begin
    variables x0(n) K(n,p)
    minimize (c'*x0)
    subject to
        A*x0+norms(A*K-B,1,2) <= b0
cvx_end

% compare 100 samples
aff_obj = []; opt_obj = [];
for i=1:100
    u = 2*rand(p,1)-1;

    cvx_begin quiet
        variable x_opt(n)
        minimize (c'*x_opt)
        subject to
            A*x_opt <= b0+B*u
```

```
    cvx_end

    aff_obj(i) = c'*(x0+K*u);
    opt_obj(i) = cvx_optval;
end

figure;
plot(opt_obj,aff_obj,'x'); hold on;
axis equal
plot(xlim, xlim);
xlabel('Optimal policy objective')
ylabel('Affine policy objective')

print -depsc affine_pol.eps
```



The (exact) expected cost of the affine policy is -3.219, the mean objective of the affine policy is -3.223 for the sample, and the mean objective of the optimal policy is -3.301 for the sample. The plot shows that the affine policy produces points that are suboptinmal, but not by much.

7. *Cost-comfort trade-off in air conditioning.* A heat pump (air conditioner) is used to cool a residence to temperature T_t in hour t, on a day with outside temperature T_t^{out}, for $t = 1, \ldots, 24$. These temperatures are given in degrees Kelvin, and we will assume that $T_t^{\text{out}} \geq T_t$.

   A total amount of heat $Q_t = \alpha(T_t^{\text{out}} - T_t)$ must be removed from the residence in hour t, where α is a positive constant (related to the quality of thermal insulation).

   The electrical energy required to pump out this heat is given by $E_t = Q_t/\gamma_t$, where

$$\gamma_t = \eta \frac{T_t}{T_t^{\text{out}} - T_t}$$

   is the *coefficient of performance* of the heat pump and $\eta \in (0, 1]$ is the efficiency constant. The efficiency is typically around 0.6 for a modern unit; the theoretical limit is $\eta = 1$. (When $T_t = T_t^{\text{out}}$, we take $\gamma_t = \infty$ and $E_t = 0$.)

   Electrical energy prices vary with the hour, and are given by $P_t > 0$ for $t = 1, \ldots, 24$. The total energy cost is $C = \sum_t P_t E_t$. We will assume that the prices are known.

   Discomfort is measured using a piecewise-linear function of temperature,

$$D_t = (T_t - T^{\text{ideal}})_+,$$

   where T^{ideal} is an ideal temperature, below which there is no discomfort. The total daily discomfort is $D = \sum_{t=1}^{24} D_t$. You can assume that $T^{\text{ideal}} < T_t^{\text{out}}$.

   To get a point on the optimal cost-comfort trade-off curve, we will minimize $C + \lambda D$, where $\lambda > 0$. The variables to be chosen are $T_1, \ldots, T_{24}$; all other quantities described above are given.

   Show that this problem has an analytical solution of the form $T_t = \psi(P_t, T_t^{\text{out}})$, where $\psi : \mathbf{R}^2 \to \mathbf{R}$. The function ψ can depend on the constants α, η, T^{ideal}, λ. Give ψ explicitly. You are free (indeed, encouraged) to check your formula using CVX, with made up values for the constants.

   Disclaimer. The focus of this course is *not* on deriving 19th century pencil and paper solutions to problems. But every now and then, a practical problem will actually have an analytical solution. This is one of them.

   Solution. We use the expression for E_t and the efficiency to get

$$E_t = (\alpha/\eta) \frac{(T_t^{\text{out}} - T_t)^2}{T_t},$$

   which is a convex function of T_t. For any practical problem we can regard the denominator as a constant, but it's cool to note that we can handle the nonlinearity of the thermodynamic efficiency exactly. It follows that the cost C, which is a positive weighted sum of E_t, is convex. The discomfort is evidently convex in T_t, so the composite objective $C + \lambda D$ is convex in T_t. So we have a convex problem here.

The composite objective $C + \lambda D$ is *separable* in T_t, *i.e.*, a sum of functions of T_t:

$$C + \lambda D = \sum_{t=1}^{24} \left(P_t(\alpha/\eta) \frac{(T_t^{\text{out}} - T_t)^2}{T_t} + \lambda (T_t - T^{\text{ideal}})_+ \right).$$

It follows that we can find each T_t (separately) by minimizing

$$P_t(\alpha/\eta) \frac{(T_t^{\text{out}} - T_t)^2}{T_t} + \lambda (T_t - T^{\text{ideal}})_+.$$

The derivative of the first term is

$$P_t(\alpha/\eta) \frac{T_t^2 - (T_t^{\text{out}})^2}{T_t^2}.$$

First assume that $T_t > T^{\text{ideal}}$. Then the optimality condition is

$$P_t(\alpha/\eta) \frac{T_t^2 - (T_t^{\text{out}})^2}{T_t^2} + \lambda = 0,$$

which gives

$$T_t = (1 + \eta\lambda/(P_t\alpha))^{-1/2} T_t^{\text{out}}.$$

If $T_t < T^{\text{ideal}}$, the optimality condition is

$$P_t(\alpha/\eta) \frac{T_t^2 - (T_t^{\text{out}})^2}{T_t^2} = 0,$$

which gives $T_t = T_t^{\text{out}}$, which contradicts $T_t < T^{\text{ideal}}$. So this case cannot happen. But we can have $T_t = T^{\text{ideal}}$; this happens when

$$(1 + \eta\lambda/(P_t\alpha))^{-1/2} T_t^{\text{out}} \leq T^{\text{ideal}}.$$

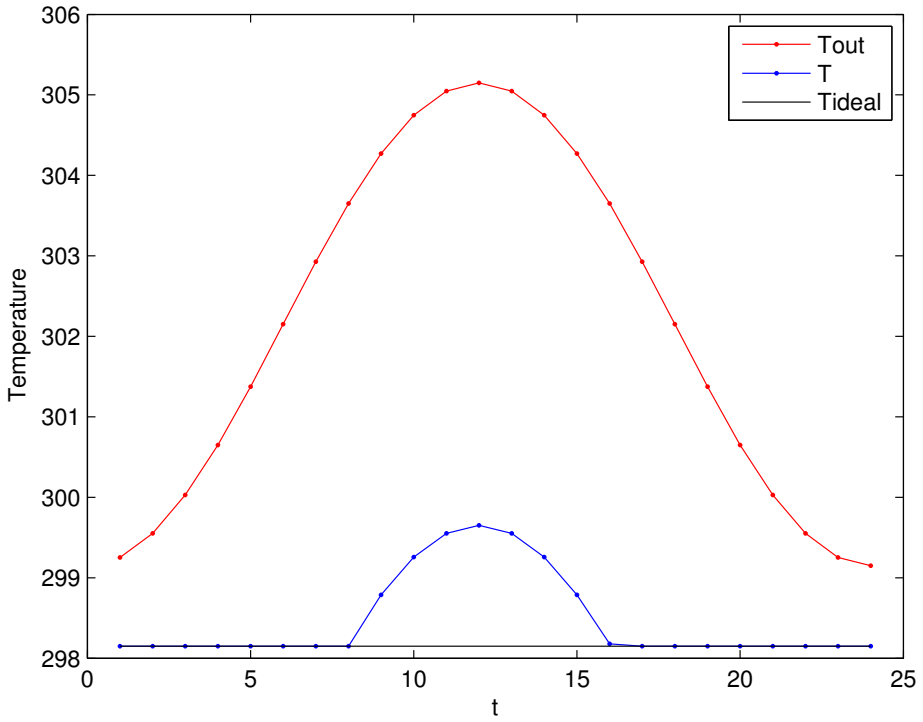
So the optimal choice of temperature is simply

$$T_t = \psi(P_t, T_t^{\text{out}}) = \max \left\{ (1 + \eta\lambda/(P_t\alpha))^{-1/2} T_t^{\text{out}}, T^{\text{ideal}} \right\}.$$

Let's test our formula using CVX.

```
% Cost-comfort trade-off in air conditioning.
N = 24;
Tout = 3*sin(2*pi*(1:N)'/24-pi/2)+29; Tideal = 25;
Tout = Tout + 273.15; Tideal = Tideal + 273.15;

eta = 0.6; alpha = 1.8; lambda = 1;
price = 6*[ones(8,1);1.5*ones(9,1);ones(7,1)];
```

```
k = price*alpha/eta;

cvx_begin
    variable T(N)
    C = k'*quad_over_lin(Tout-T,T,2);
    D = sum(pos(T-Tideal));
    minimize ( C + lambda*D )
    subject to
        T >= 0; T <= Tout; % they are not necessary
cvx_end

T_analytic = max(sqrt(k./(k+lambda)).*Tout, Tideal);
C_analytic = k'*quad_over_lin(Tout-T_analytic,T_analytic,2);
display(['Total cost obtained by cvx: ' num2str(C)]);
display(['Total cost obtained analytically: ' num2str(C_analytic)]);

plot((1:N)',Tout,'.-r',(1:N)',T,'.-b',(1:N)',Tideal*ones(N,1),'-k');
xlabel('t'); ylabel('Temperature'); legend('Tout','T','Tideal');

print -depsc air_cond.eps
```



The total energy cost obtained from the analytical solution is 31.838. CVX returns the same answer (31.838, when using sdpt3; and 31.8204, when using Sedumi).

8. *Least-cost road grading.* A road is to be built along a given path. We must choose the height of the roadbed (say, above sea level) along the path, minimizing the total cost of grading, subject to some constraints. The cost of grading (*i.e.*, moving earth to change the height of the roadbed from the existing elevation) depends on the difference in height between the roadbed and the existing elevation. When the roadbed is below the existing elevation it is called a *cut*; when it is above it is called a *fill*. Each of these incurs engineering costs; for example, fill is created in a series of *lifts*, each of which involves dumping just a few inches of soil and then compacting it. Deeper cuts and higher fills require more work to be done on the road shoulders, and possibly, the addition of reinforced concrete structures to stabilize the earthwork. This explains why the marginal cost of cuts and fills increases with their depth/height.

   We will work with a discrete model, specifying the road height as h_i, $i = 1, \ldots, n$, at points equally spaced a distance d from each other along the given path. These are the variables to be chosen. (The heights $h_1, \ldots, h_n$ are called a *grading plan*.) We are given e_i, $i = 1, \ldots, n$, the existing elevation, at the points. The grading cost is

$$C = \sum_{i=1}^{n} \left(\phi^{\text{fill}}((h_i - e_i)_+) + \phi^{\text{cut}}((e_i - h_i)_+) \right),$$

   where ϕ^{fill} and ϕ^{cut} are the fill and cut cost functions, respectively, and $(a)_+ = \max\{a, 0\}$. The fill and cut functions are increasing and convex. The goal is to minimize the grading cost C.

   The road height is constrained by given limits on the first, second, and third derivatives:

$$\begin{aligned} |h_{i+1} - h_i|/d &\leq D^{(1)}, \quad i = 1, \ldots, n-1 \\ |h_{i+1} - 2h_i + h_{i-1}|/d^2 &\leq D^{(2)}, \quad i = 2, \ldots, n-1 \\ |h_{i+1} - 3h_i + 3h_{i-1} - h_{i-2}|/d^3 &\leq D^{(3)}, \quad i = 3, \ldots, n-1, \end{aligned}$$

   where $D^{(1)}$ is the maximum allowable road slope, $D^{(2)}$ is the maximum allowable curvature, and $D^{(3)}$ is the maximum allowable third derivative.

   (a) Explain how to find the optimal grading plan.

   (b) Find the optimal grading plan for the problem with data given in `road_grading_data.m`, and fill and cut cost functions

$$\phi^{\text{fill}}(u) = 2(u)_+^2 + 30(u)_+, \qquad \phi^{\text{cut}} = 12(u)_+^2 + (u)_+.$$

   Plot $h_i - e_i$ for the optimal grading plan and report the associated cost.

   (c) Suppose the optimal grading problem with $n = 1000$ can be solved on a particular machine (say, with one, or just a few, cores) in around one second. Assuming the author of the software took EE364a, about how long will it take to solve the optimal grading problem with $n = 10000$? Give a very brief justification of your answer, no more than a few sentences.

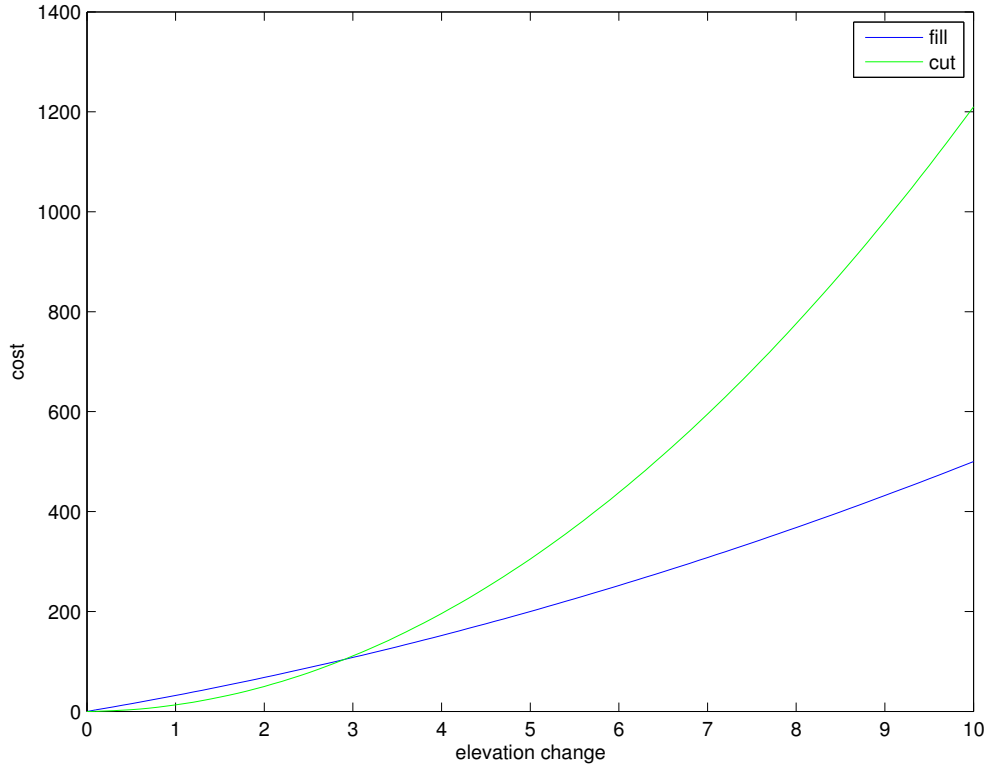
Solution.

(a) Fortunately, this problem in convex; C is convex since ϕ^{fill} and ϕ^{cut} are convex and increasing and max is convex. Therefore we simply solve the problem

$$\begin{array}{ll} \text{minimize} & C \\ \text{subject to} & |h_{i+1} - h_i|/d \leq D^{(1)}, \quad i = 1, \ldots, n-1 \\ & |h_{i+1} - 2h_i + h_{i-1}|/d^2 \leq D^{(2)}, \quad i = 2, \ldots, n-1 \\ & |h_{i+1} - 3h_i + 3h_{i-1} - h_{i-2}|/d^3 \leq D^{(3)}, \quad i = 3, \ldots, n-1, \end{array}$$

with optimization variables h_i.

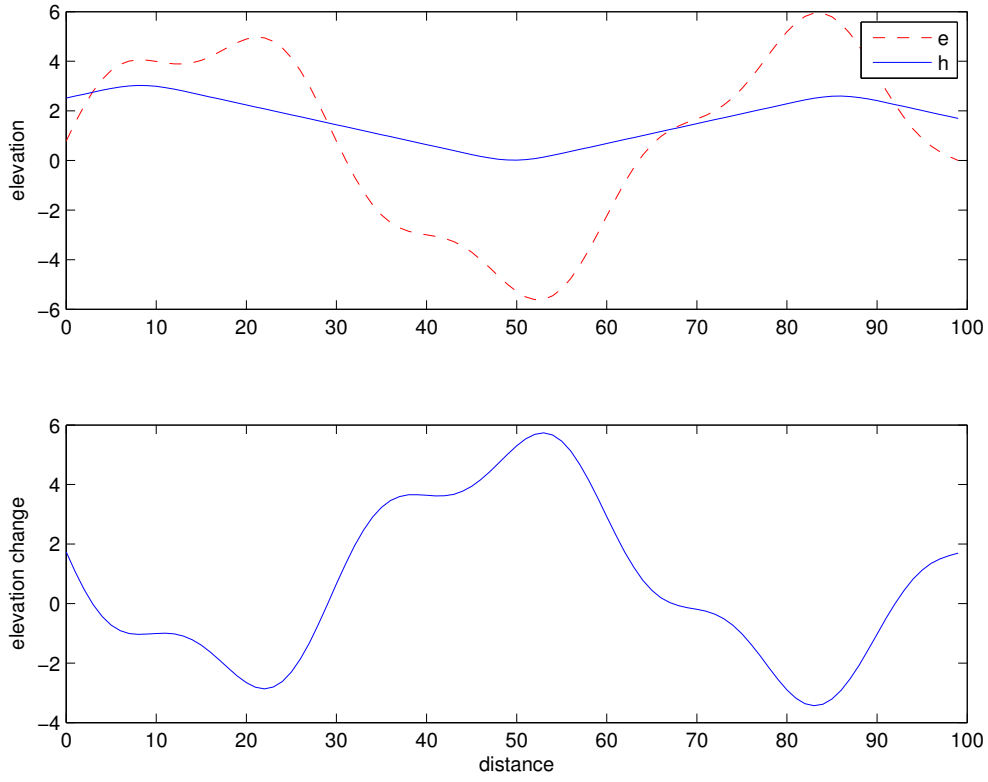
(b) For this problem instance we have cost functions shown below.



The following code solves the problem:

```
% Least-cost road grading.
road_grading_data;

cvx_begin
variables h(n);
minimize sum(alpha_fill*square_pos(h-e)+beta_fill*pos(h-e)+...
     alpha_cut*square_pos(e-h)+beta_cut*pos(e-h))
subject to
```

```
abs(h(2:n)-h(1:n-1)) <= D1*d
abs(h(3:n)-2*h(2:n-1)+h(1:n-2)) <= D2*d^2
abs(-h(4:n)+3*h(3:n-1)-3*h(2:n-2)+h(1:n-3)) <= D3*d^3
cvx_end

figure
subplot(2,1,1)
plot((0:n-1)*d,e,'--r');
ylabel('elevation');
hold on
plot((0:n-1)*d,h, 'b')
legend('e','h');
subplot(2,1,2)
plot((0:n-1)*d,h-e)
ylabel('elevation change')
xlabel('distance')

print -depsc road_grading;
```



We find that the optimal cost is 7562.82.

(c) Using our knowledge from EE364a we see immediately that this problem is well structured. Our objective is separable and our constraints have bandwidth of at

most 4. For each Newton step we have to solve a banded system, which we know we can do in $\mathcal{O}(nk^2)$ flops, where $k = 4$ is the bandwidth. We can, therefore, take a Newton step in $\mathcal{O}(n)$ flops. We have seen that the number of iterations required to solve a problem with an interior point method is practically independent of problem size. Thus, if an optimal grading problem with $n = 1000$ can be solved in about 1 second, we can solve a problem with $n = 10000$ in approximately 10 seconds.

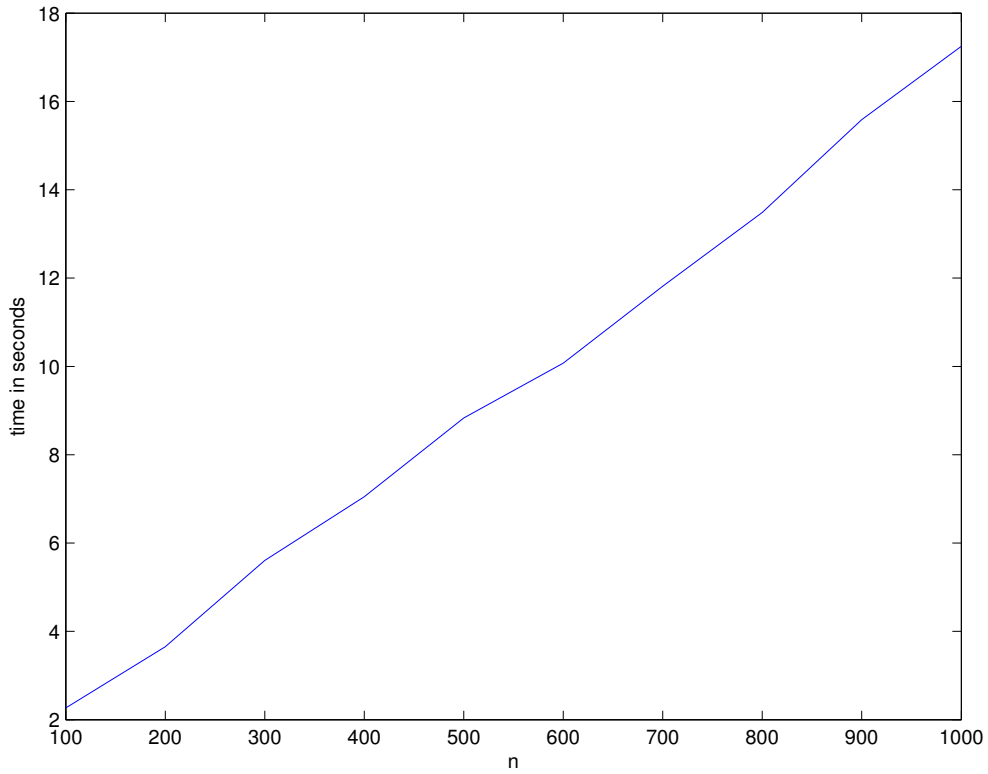
To see how this banded system arises, using EE364a knowledge you might solve this problem using an interior point barrier method. For a given t you are now solving the unconstrained minimization problem

$$\begin{array}{ll}\text{minimize} & tC - \sum_{i=1}^{n-1}\left(\log(D^{(1)}d - h_{i+1} + h_i) + \log(D^{(1)}d + h_{i+1} - h_i)\right) - \\ & \sum_{i=2}^{n-1}\left(\log(D^{(2)}d^2 - h_{i+1} + 2h_i - h_{i-1})\right) - \\ & \sum_{i=2}^{n-1}\left(\log(D^{(2)}d^2 + h_{i+1} - 2h_i + h_{i-1})\right) - \\ & \sum_{i=3}^{n-1}\left(\log(D^{(3)}d^3 - h_{i+1} + 3h_i - 3h_{i-1} + h_{i-2})\right) - \\ & \sum_{i=3}^{n-1}\left(\log(D^{(3)}d^3 + h_{i+1} - 3h_i + 3h_{i-1} - h_{i-2})\right).\end{array}$$

We can represent the Hessian of this function as the sum of 4 matrices: the Hessian relating to C and the Hessians relating to constraints with $D^{(1)}$, $D^{(2)}$, and $D^{(3)}$. Since C is separable in the optimization variables h_i, its Hessian is diagonal. The constraints with $D^{(1)}$ will contribute a matrix of bandwidth 2 to the Hessian, as there is only coupling between h_i and h_{i+1}. Similarly the terms related to $D^{(2)}$ will contribute a matrix of bandwidth 3, and the terms related to $D^{(3)}$ will contribute a matrix of bandwidth 4. Therefore at each Newton step, as already stated we must solve a system with bandwidth 4. As the problem size increases, the bandwidth remains constant, so we expect the problem to scale linearly in n until we run into system related issues such as memory limitations. The code below generates problem instances from $n = 100$ to $n = 1000$:

```
% Least-cost road grading timing for different problem sizes.
road_grading_data

N = 10;
times = zeros(N,1);
e2 = [];
for i = 1:N
    e2 = [e2; e];
    tic
    cvx_begin
    variables h(i*n);
    minimize sum(alpha_fill*square_pos(h-e2)+...
     beta_fill*pos(h-e2)+...
     alpha_cut*square_pos(e2-h)+beta_cut*pos(e2-h))
```

```
    subject to
    abs(h(2:end)-h(1:end-1)) <= D1*d
    abs(h(3:end)-2*h(2:end-1)+h(1:end-2)) <= D2*d^2
    abs(-h(4:end)+3*h(3:end-1)-3*h(2:end-2)+h(1:end-3)) <= D3*d^3
    cvx_end
    times(i) = toc;
end

figure
plot((1:N)*n,times);
xlabel('n');
ylabel('time in seconds');

print -depsc road_grading_timing.eps
```



We see that in SDPT3 the timing is indeed linear with problem size (with an offset for the CVX overhead).