Homework 3 Solutions

EE 263 Stanford University

Summer 2017

1. Fitting a model for hourly temperature. You are given a set of temperature measurements (in degrees C), $y_t \in \mathbb{R}$, $t = 1, \ldots, N$, taken hourly over one week (so $N = 168$). An expert says that over this week, an appropriate model for the hourly temperature is a trend (*i.e.*, a linear function of t) plus a diurnal component (*i.e.*, a 24-periodic component):

$$\hat{y}_t = at + p_t,$$

where $a \in \mathbb{R}$ and $p \in \mathbb{R}^N$ satisfies $p_{t+24} = p_t$, for $t = 1, \ldots, N - 24$. We can interpret a (which has units of degrees C per hour) as the warming or cooling trend (for $a > 0$ or $a < 0$, respectively) over the week.

a) Explain how to find $a \in \mathbb{R}$ and $p \in \mathbb{R}^N$ (which is 24-periodic) that minimize the RMS value of $y - \hat{y}$.

b) Carry out the procedure described in part (a) on the data set found in `tempfit_data.m`.

Give the value of the trend parameter a that you find. Plot the model $\hat{y}$ and the measured temperatures y on the same plot. (The matlab code to do this is in the data file, but commented out.)

c) *Temperature prediction.* Use the model found in part (b) to predict the temperature for the next 24-hour period (*i.e.*, from $t = 169$ to $t = 192$). The file `tempdata.m` also contains a 24 long vector `ytom` with tomorrow's temperatures. Plot tomorrow's temperature and your prediction of it, based on the model found in part (b), on the same plot. What is the RMS value of your prediction error for tomorrow's temperatures?

Solution.

a) Since p is 24-periodic, we only need to specify its values for $t = 1, \ldots, 24$. We can express the vector of model temperatures as

$$\hat{y} = Ax,$$

where

$$x = \begin{bmatrix} a \\ p_1 \\ \vdots \\ p_{24} \end{bmatrix} \in \mathbb{R}^{25}, \quad A = \begin{bmatrix} 1 & I_{24\times 24} \\ 2 & \\ \vdots & \vdots \\ 168 & I_{24\times 24} \end{bmatrix} \in \mathbb{R}^{168\times 25},$$

where the righthand part of A consists of 7 24×24 identity matrices stacked on op of each other.

The solution is given by

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

The associated model is given by $\hat{y} = Ax$.

This is a perfectly acceptable answer. But in this case we can work out a more explicit solution. We did not expect any of you to do this, and we don't even consider this a better solution. But we mention it anyway.

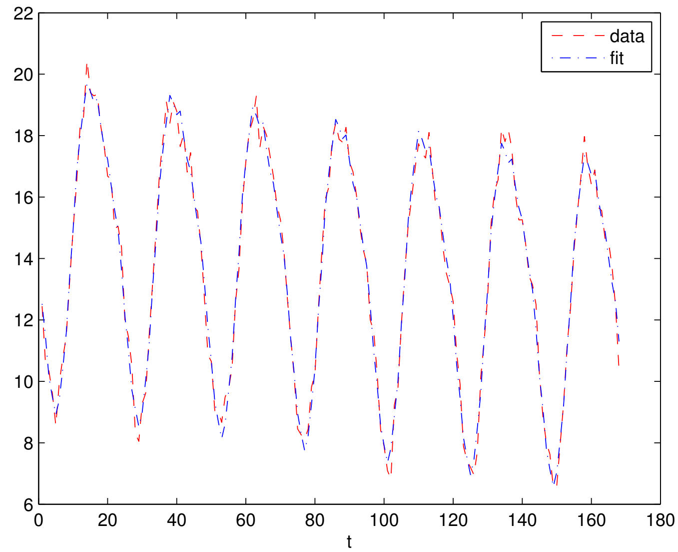
First, we compute the value of a. Using the periodicity of p we have $y_t - at = y_{t+24k} - a(t + 24k)$, $k = 0, \ldots, 6$. We solve for a and we have that $a = (y_{t+24k} - y_t)/(24k)$. We compute the value of a for $k = 0, \ldots, 6$ and $t = 1, \ldots, 24$, and we get the mean as the estimate of a. Now, let's find the values of p. We know that $p_t = p_{t+24k}$, for $k = 0, \ldots, 6$ and $t = 1, \ldots, 24$. By taking the average value of p_t over the period of seven days we find an estimate of p_t, $t = 1, \ldots, 24$,

$$p_t = \frac{1}{7} \sum_{k=0}^{6} (y_{24k+t} - a(24k + t)),$$

b) The following matlab script solves part (b) and part (c).

```
% loading the data from the files given
tempdata
% matrix A of the system y=Ax
e24=eye(24);
A = [ e24; e24; e24; e24; e24; e24; e24];
A = [(1:N)' A];
% estimate coeffs
x = A\y ;
% the fit
yhat = (A*x)';
% plot the fit and the data on the same figure
plot([1:168],y,'--r',[1:168],yhat,'-.');
% now we solve part (c)
% prediction of tomorrow's temperature using the fit
ytomhat = ([(N+1:N+24)' e24]*x)';
% plot the fit and the data for next day
figure;
plot([1:24],ytom,'r',[1:24],ytomhat,'-.');
%RMS error
RMS = sqrt(norm(ytomhat-ytom)^2/24)
```

By running the above script we get $a = -0.0121$. The fit, $\hat{y}$, along with the data, y, are presented in the following figure.



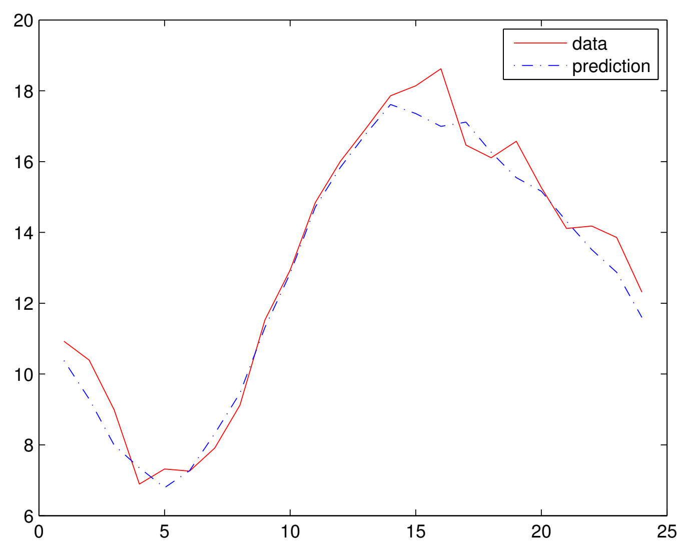
c) In part (b) we estimated the values of a and p_i, $i = 1, \dots, 24$, collected together as one vector $x \in \mathbb{R}^{25}$. To estimate y_{tom}, the 24-vector of tomorrow's temperatures, we use

$$y_{\text{tom}} = A_{\text{tom}} x,$$

where x is the value found in part (b), and

$$A_{\text{tom}} = \begin{bmatrix} 169 & \\ 170 & \\ \vdots & I_{24\times 24} \\ 192 & \end{bmatrix}.$$

The prediction of the temperature along with the data for the next day are presented in the following figure.



The RMS value of the difference between the prediction and the data for the next day is 0.6522.

2. **Curve-smoothing.** We are given a function $F : [0,1] \to \mathbb{R}$ (whose graph gives a curve in $\mathbb{R}^2$). Our goal is to find another function $G : [0,1] \to \mathbb{R}$, which is a *smoothed* version of F. We'll judge the smoothed version G of F in two ways:

   - *Mean-square deviation from F,* defined as
   $$D = \int_0^1 (F(t) - G(t))^2 \, dt.$$
   - *Mean-square curvature,* defined as
   $$C = \int_0^1 G''(t)^2 \, dt.$$

   We want *both* D and C to be small, so we have a problem with two objectives. In general there will be a trade-off between the two objectives. At one extreme, we can choose $G = F$, which makes $D = 0$; at the other extreme, we can choose G to be an affine function (*i.e.*, to have $G''(t) = 0$ for all $t \in [0,1]$), in which case $C = 0$. The problem is to identify the optimal trade-off curve between C and D, and explain how to find smoothed functions G on the optimal trade-off curve. To reduce the problem to a finite-dimensional one, we will represent the functions F and G (approximately) by vectors $f, \ g \in \mathbb{R}^n$, where
   $$f_i = F(i/n), \quad g_i = G(i/n).$$
   You can assume that n is chosen large enough to represent the functions well. Using this representation we will use the following objectives, which approximate the ones defined for the functions above:

   - *Mean-square deviation,* defined as
   $$d = \frac{1}{n} \sum_{i=1}^{n} (f_i - g_i)^2.$$
   - *Mean-square curvature,* defined as
   $$c = \frac{1}{n-2} \sum_{i=2}^{n-1} \left(\frac{g_{i+1} - 2g_i + g_{i-1}}{1/n^2} \right)^2.$$

   In our definition of c, note that
   $$\frac{g_{i+1} - 2g_i + g_{i-1}}{1/n^2}$$
   gives a simple approximation of $G''(i/n)$. You will only work with this approximate version of the problem, *i.e.*, the vectors f and g and the objectives c and d.

   a) Explain how to find g that minimizes $d + \mu c$, where $\mu \geq 0$ is a parameter that gives the relative weighting of sum-square curvature compared to sum-square deviation. Does your method always work? If there are some assumptions you need to make (say, on rank of some matrix, independence of some vectors, etc.), state them clearly. Explain how to obtain the two extreme cases: $\mu = 0$, which corresponds to minimizing d without regard for c, and also the solution obtained as $\mu \to \infty$ (*i.e.*, as we put more and more weight on minimizing curvature).

b) Get the file `curve_smoothing.m` from the course web site. This file defines a specific vector f that you will use. Find and plot the optimal trade-off curve between d and c. Be sure to identify any critical points (such as, for example, any intersection of the curve with an axis). Plot the optimal g for the two extreme cases $\mu = 0$ and $\mu \to \infty$, and for three values of μ in between (chosen to show the trade-off nicely). On your plots of g, be sure to include also a plot of f, say with dotted line type, for reference. Submit your matlab code.

Solution.

a) Let's start with the two extreme cases. When $\mu = 0$, finding g to minimize $d + \mu c$ reduces to finding g to minimize d. Since d is a sum of squares, $d \geq 0$. Choosing $g = f$ trivially achieves $d = 0$. This makes perfect sense: to minimize the deviation measure, just take the smoothed version to be the same as the original function. This yields zero deviation, naturally, but also, it yields no smoothing! Next, consider the extreme case where $\mu \to \infty$. This means we want to make the curvature as small as possible. Can we drive it to zero? The answer is yes, we can: the curvature is zero if and only if g is an affine function, *i.e.*, has the form $g_i = ai + b$ for some constants a and b. There are lots of vectors g that have this form; in fact, we have one for every pair of numbers a, b. All of these vectors g make c zero. Which one do we choose? Well, even if μ is huge, we still have a small contribution to $d + \mu c$ from d, so among all g that make $c = 0$, we'd like the one that minimizes d. Basically, we want to find the best affine approximation, in the sum of squares sense, to f. We want to find a and b that minimize

$$\left\| f - A \begin{bmatrix} a \\ b \end{bmatrix} \right\| \text{ where } A = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ \vdots & \vdots \\ n & 1 \end{bmatrix}.$$

For $n \geq 2$, A is skinny and full rank, and a and b can be found using least-squares. Specifically, $[a\ b]^\mathsf{T} = (A^\mathsf{T} A)^{-1} A^\mathsf{T} f$. In the general case, minimizing $d + \mu c$, is the same as choosing g to minimize

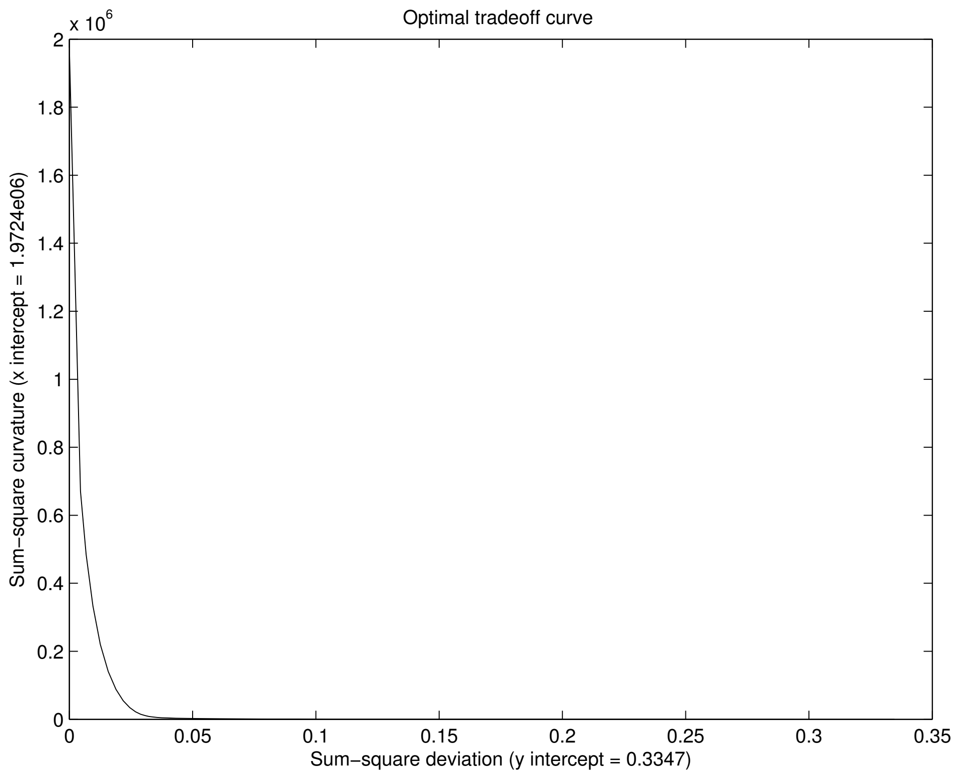
$$\left\| \frac{1}{\sqrt{n}} I g - \frac{1}{\sqrt{n}} f \right\|^2 + \mu \left\| \underbrace{\frac{n^2}{\sqrt{n-2}} \begin{bmatrix} -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 & 2 & -1 \end{bmatrix}}_{S} g \right\|^2.$$

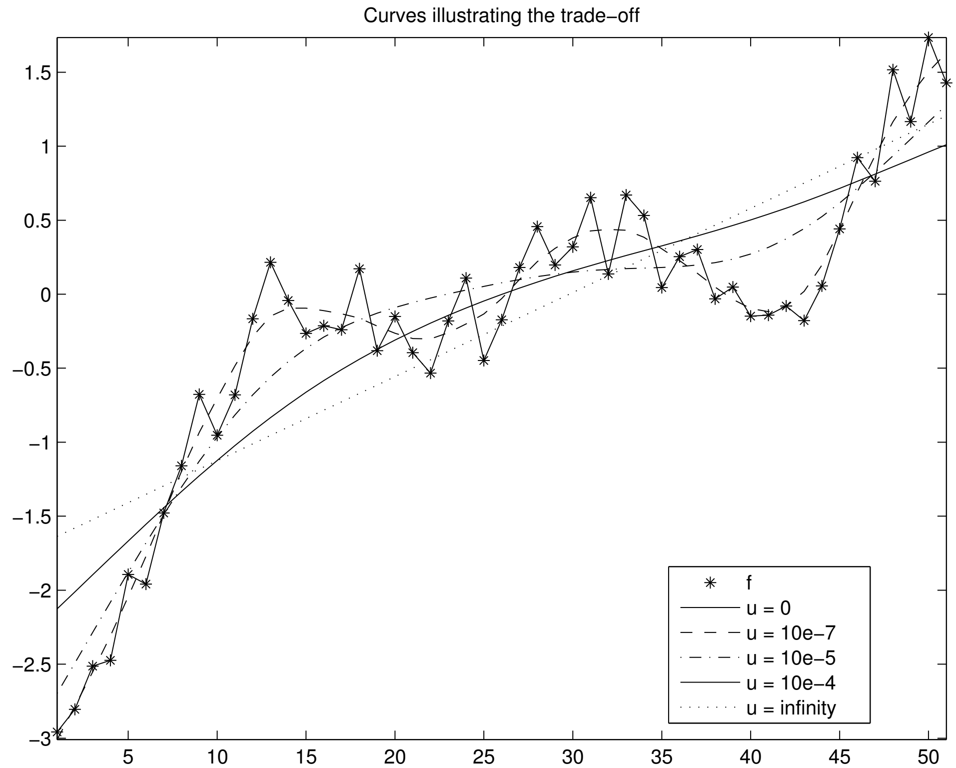
This is a multi-objective least-squares problem. The minimizing g is

$$g = (\tilde{A}^{\mathsf{T}}\tilde{A})^{-1}\tilde{A}^{\mathsf{T}}\tilde{y} \text{ where } \tilde{A} = \begin{bmatrix} \frac{I}{\sqrt{n}} \\ \sqrt{\mu}S \end{bmatrix} \text{ and } \tilde{y} = \begin{bmatrix} \frac{f}{\sqrt{n}} \\ 0 \end{bmatrix}.$$

The inverse of $\tilde{A}$ always always exists because I is full rank. The expression can also be written as $g = (\frac{I}{n} + \mu S^{\mathsf{T}}S)^{-1}\frac{f}{n}$.

b) The following plots show the optimal trade-off curve and the optimal g corresponding to representative μ values on the curve.





The following matlab code finds and plots the optimal trade-off curve between d and c. It also finds and plots the optimal g for representative values of μ. As expected, when $\mu = 0$, $g = f$ and no smoothing occurs. At the other extreme, as μ goes to infinity, we get an affine approximation of f. Intermediate values of μ correspond to approximations of f with different degrees of smoothness.

```
close all;
clear all;
curve_smoothing
S = toeplitz([-1; zeros(n-3,1)],[-1 2 -1 zeros(1,n-3)]);
S = S*n^2/(sqrt(n-2));
I = eye(n);
g_no_deviation = f;
error_curvature(1) = norm(S*g_no_deviation)^2;
error_deviation(1) = 0;
u = logspace(-8,-3,30);
for i = 1:length(u)
A_tilde = [1/sqrt(n)*I; sqrt(u(i))*S];
y_tilde = [1/sqrt(n)*f; zeros(n-2,1)];
g = A_tilde\y_tilde;
error_deviation(i+1) = norm(1/sqrt(n)*I*g-f/sqrt(n))^2;
error_curvature(i+1) = norm(S*g)^2;
end
a1 = 1:n;
```

```
a1 = a1';
a2 = ones(n,1);
A = [a1 a2];
affine_param = inv(A'*A)*A'*f;
for i = 1:n
g_no_curvature(i) = affine_param(1)*i+affine_param(2);
end
g_no_curvature = g_no_curvature';
error_deviation(length(u)+2) = 1/n*norm(g_no_curvature-f)^2;
error_curvature(length(u)+2) = 0;
figure(1);
plot(error_deviation, error_curvature);
xlabel('Sum-square deviation (y intercept = 0.3347)');
ylabel('Sum-square curvature (x intercept = 1.9724e06)');
title('Optimal tradeoff curve ');
print curve_extreme.eps;
u1 = 10e-7;
A_tilde = [1/sqrt(n)*I;sqrt(u1)*S];
y_tilde = [1/sqrt(n)*f;zeros(n-2,1)];
g1 = A_tilde\y_tilde;
u2 = 10e-5;
A_tilde = [1/sqrt(n)*I;sqrt(u2)*S];
y_tilde = [1/sqrt(n)*f;zeros(n-2,1)];
g2 = A_tilde\y_tilde;
u3 = 10e-4;
A_tilde = [1/sqrt(n)*I;sqrt(u3)*S];
y_tilde = [1/sqrt(n)*f;zeros(n-2,1)];
g3 = A_tilde\y_tilde;
figure(3);
plot(f,'*');
hold;
plot(g_no_deviation);
plot(g1,'--');
plot(g2,'-.');
plot(g3,'-');
plot(g_no_curvature,':');
axis tight;
legend('f','u = 0','u = 10e-7', 'u = 10e-5', 'u = 10e-4','u = infinity',0);
title('Curves illustrating the trade-off');
print curve_tradeoff.eps;
```

Note: Several exams had a typo that defined

$$c = \frac{1}{n-1}\sum_{i=2}^{n-1}\left(\frac{g_{i+1}-2g_i+g_{i-1}}{1/n^2}\right)^2$$

instead of

$$c = \frac{1}{n-2} \sum_{i=2}^{n-1} \left(\frac{g_{i+1} - 2g_i + g_{i-1}}{1/n^2} \right)^2 .$$

The solutions above reflect the second definition. Full credit was given for answers consistent with either definition. *Some common errors*

- Several students tried to approximate f using low-degree polynomials. While fitting f to a polynomial does smooth f, it does not necessarily minimize $d + \mu c$ for some $\mu \geq 0$, nor does it illustrate the trade-off between curvature and deviation.
- In explaining how to find the g that minimizes $d + \mu c$ as $\mu \to \infty$, many people correctly observed that if $g \in \text{null}(S)$, then $c = 0$. For full credit, however, solutions had to show how to choose the vector in null(S) that minimizes d.
- Many people chose to zoom in on a small section of the trade-off curve rather than plot the whole range from 0 to $\mu \to \infty$. Those solutions received full-credit provided they calculated the intersections with the axes (i.e. provided they found the minimum value for $d + \mu c$ when $\mu = 0$ and when $\mu \to \infty$).

3. **Hovercraft with limited range.** We have a hovercraft moving in the plane with two thrusters, each pointing through the center of mass, exerting forces in the **x** and **y** directions with 100% efficiency. The hovercraft has mass 1. The discretized equations of motion for the hovercraft are

$$x(t+1) = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} \frac{1}{2} & 0 \\ 1 & 0 \\ 0 & \frac{1}{2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

where x_1 and x_2 are the position and velocity in the **x**-direction, and x_3, x_4 are the position and velocity in the **y**-direction. Here

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}$$

is the force acting on the hovercraft for time in the interval $[t, t+1)$. Let the position of the vehicle at time t be $q(t) \in \mathbb{R}^2$.

a) The hovercraft starts at the origin. We'd like to apply thrust to make it move through points p_1, p_2, p_3 at times t_1, t_2, t_3, where

$$p_1 = \begin{bmatrix} 1 \\ -\frac{1}{2} \end{bmatrix} \qquad p_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad p_3 = \begin{bmatrix} -\frac{3}{2} \\ 0 \end{bmatrix}$$

$$t_1 = 6 \qquad t_2 = 40 \qquad t_3 = 50$$

We will run the hovercraft on the time interval $[0, 70]$. We'd like to apply a sequence of inputs $u(0), u(1), \ldots, u(70)$ to make the hovercraft position pass through the above sequence of points at the specified times.

We would like to find the sequence of inputs that drives the hovercraft through the desired points which has the minimum cost, given by the sum of the squares of the forces:

$$\sum_{t=0}^{70} \|u(t)\|^2$$

To do this, pick A_{hov} and y_{des} to set this problem up as an equivalent minimum-norm problem, where we would like to find the minimum-norm u_{seq} which satisfies

$$A_{\text{hov}} u_{\text{seq}} = y_{\text{des}}$$

where u_{seq} is the sequence of force inputs

$$u_{\text{seq}} = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(70) \end{bmatrix}$$

Plot the trajectory of the hovercraft using this input, and the way-points $p_1, \ldots, p_3$. Also plot the optimal u against time.

b) Now we would like to compute the trade-off curve between the accuracy with which the mass passes through the waypoints and the norm of the force used. Let our two objective functions be

$$J_1 = \sum_{i=1}^{3} \|q(t_i) - p_i\|^2 = \|A_{\text{hov}} u_{\text{seq}} - y_{\text{des}}\|^2$$

and

$$J_2 = \sum_{t=0}^{70} \|u(t)\|^2$$

By minimizing the weighted sum

$$J_1 + \mu J_2$$

for a range of values of μ, plot the trade-off curve of J_1 against J_2 showing the achievable performance. To generate suitable values of μ, you may find the `logspace` command useful in Matlab; you'll need to pick appropriate maximum and minimum values. This above trade-off curve shows how we can trade-off between how accurately the hovercraft passes through the waypoints and how much input energy is used.

c) For each of the following values of μ

$$\{\, 10^{\frac{p}{2}} \mid p = -2, 0, 2, \ldots, 10 \,\}$$

plot the trajectories all on the same plot, together with the waypoints.

d) Now suppose we are controlling the hovercraft by radio control, and the maximum range possible between the transmitter and receiver is 2 (in whatever units we are using for distance.) Notice that, if we use the minimum-norm input then the hovercraft passes out of range, both when making its first turn and on the final stretch (between times 50 and 70).

We'd like to do something about this, but trading off the input norm as above doesn't do the right thing; if μ is large then the hovercraft stays within range, but misses the waypoints entirely; if μ is small then it comes close to the waypoints, but goes out of range. Notice that this is particularly a problem on the final stretch between times 50 and 70; explain why this is.

e) One remedy for this problem is to solve a *constrained multiobjective least-squares* problem. We would like to impose the constraint that

$$A_{\text{hov}} u_{\text{seq}} = y_{\text{des}}$$

that is, achieve zero waypoint error $J_1 = 0$. We can attempt to keep the hovercraft in range by trading off the sum of the squares of the *position*

$$J_3 = \sum_{t=0}^{70} \|q(t)\|^2$$

against input cost J_2 subject to this constraint. To do this, we'll solve

$$\begin{array}{ll} \text{minimize} & J_3 + \gamma J_2 \\ \text{subject to} & A_{\text{hov}} u_{\text{seq}} = y_{\text{des}} \end{array}$$

First, find the matrix W so that the cost function is given by

$$J_3 + \gamma J_2 = \|W u_{\text{seq}}\|^2$$

f) Now we have a problem of the form

$$\begin{array}{ll} \text{minimize} & \|Wu\|^2 \\ \text{subject to} & Au = y_{\text{des}} \end{array}$$

This is called a *weighted minimum-norm solution*; the only difference from the usual minimum-norm solution to $Au = y_{\text{des}}$ is the presence of the matrix W, and when $W = I$ the optimal u is just given by $u_{\text{opt}} = A^{\dagger} y_{\text{des}}$. Show that the solution for general W is

$$u_{\text{opt}} = \Sigma^{-1} A^T (A \Sigma^{-1} A^T)^{-1} y_{\text{des}}$$

where $\Sigma = W^T W$. (One way to do this is using Lagrange multipliers.) Use this to solve the remaining parts of this problem.

g) For each of the following values of γ

$$\{\, 10^{\frac{p}{2}} \mid p = 0, 2, 4, \ldots, 20 \,\}$$

Plot the trajectories all on the same plot, together with the waypoints. Explain what you see.

h) By trying different values of γ, you should be able to find a trajectory which just keeps the hovercraft within range. Plot the trajectory of the hovercraft; what is the corresponding value of γ? Is this the smallest-norm input u that just keeps the hovercraft within range, and drives the hovercraft through the waypoints? Explain why, or why not.

i) For a range of values of γ, plot the trade-off curve of J_3 against J_2 showing the achievable performance.

Solution.

a) Setting

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

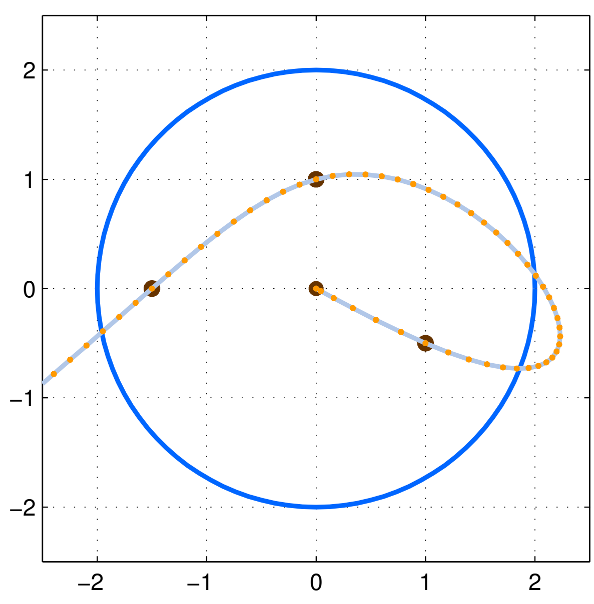
gives the position of the hovercraft at time t as

$$y(t) = \sum_{\tau=0}^{t-1} CA^{t-1-\tau}Bu(\tau)$$

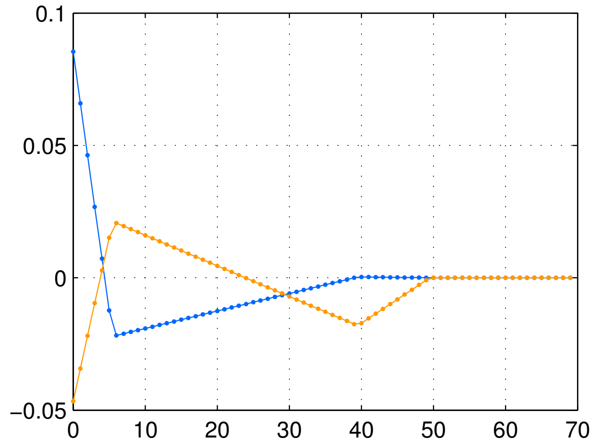
The parameters for the least-squares problem are therefore

$$A_{\text{hov}} = \begin{bmatrix} CA^{t_1-1}B & CA^{t_1-1} & \cdots & CB & 0 & 0 & \ldots & 0 \\ CA^{t_2-1}B & CA^{t_2-2}B & & & \cdots & & & 0 \\ CA^{t_3-1}B & CA^{t_3-2}B & & & \cdots & & & 0 \end{bmatrix} \qquad y_{\text{des}} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

Solving this least squares problem gives optimal trajectory



The corresponding optimal input sequence is below.



b) The weighted sum objective is

$$J_1 + \mu J_2 = \left\| \begin{bmatrix} A_{\text{hov}} \\ \sqrt{\mu} I \end{bmatrix} u_{\text{seq}} - \begin{bmatrix} y_{\text{des}} \\ 0 \end{bmatrix} \right\|^2$$

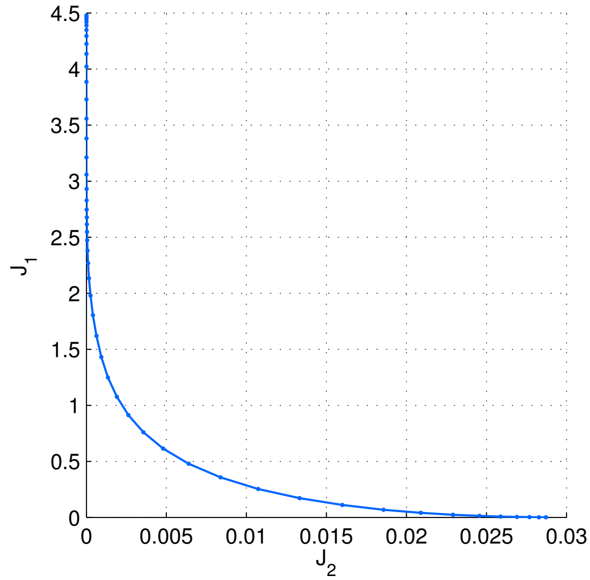
where

$$u_{\text{seq}} = \begin{bmatrix} u(0) \\ \vdots \\ u(69) \end{bmatrix}$$

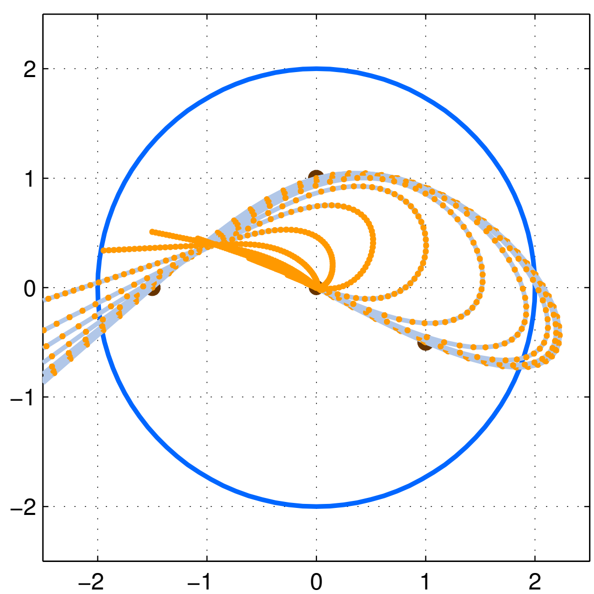
and so the optimal input sequence is given by

$$u_{\text{seq}} = \begin{bmatrix} A_{\text{way}} \\ \sqrt{\mu} I \end{bmatrix}^{\dagger} \begin{bmatrix} y_{\text{des}} \\ 0 \end{bmatrix}$$

Choosing values of μ between 1 and 10^7 using `mus=logspace(0,7,50)`, the trade-off curve is shown below.



c) All of the trajectories together are



We can see clearly that increasing μ reduces the accuracy with which the trajectory passes through the waypoints.

d) On the final stretch the input is zero, and so is unaffected by increasing μ. We were attempting to use the heuristic 'keeping u small keeps x small' but this fails, because when $u = 0$ the hovercraft just keeps going in a straight line.

e) We would like to minimize $J_3 + \gamma J_2$ subject to the constraints that the hovercraft moves through the waypoints. Denote the sequence of positions of the hovercraft by

$$y_{\text{seq}} = \begin{bmatrix} y(0) \\ \vdots \\ y(T) \end{bmatrix}$$

where $T = 70$. Then we have

$$y_{\text{seq}} = T u_{\text{seq}}$$

where T is the Toeplitz matrix

$$T = \begin{bmatrix} 0 & & & \\ CB & 0 & & \\ CAB & CB & 0 & \\ \vdots & & \ddots & \\ CA^{T-1}B & CA^{T-2}B & \dots & CB \end{bmatrix}$$

Now the cost function is

$$\begin{aligned} J_3 + \gamma J_2 &= \|T u_{\text{seq}}\|^2 + \gamma \|u_{\text{seq}}\|^2 \\ &= \|W u_{\text{seq}}\|^2 \end{aligned}$$

where

$$W = \begin{bmatrix} T \\ \sqrt{\gamma} I \end{bmatrix}$$

f) We'd like to solve

$$\begin{array}{ll} \text{minimize} & \|Wu\|^2 \\ \text{subject to} & Au = y_{\text{des}} \end{array}$$

One way to solve this is using Lagrange multipliers; if we augment the cost function by the Lagrange multipliers multiplied by the constraints, we have

$$L(u, \lambda) = u^T \Sigma u + \lambda^T (Au - y_{\text{des}})$$

and the optimality conditions are

$$\begin{aligned} \frac{\partial L}{\partial u} &= 2u_{\text{opt}}^T \Sigma + \lambda^T A = 0 \\ \frac{\partial L}{\partial \lambda} &= u_{\text{opt}}^T A^T - y_{\text{des}}^T = 0 \end{aligned}$$

The first condition gives

$$u_{\text{opt}} = -\frac{1}{2} \Sigma^{-1} A^T \lambda$$

and substituting this into the second we have

$$-\frac{1}{2} A \Sigma^{-1} A^T \lambda = y_{\text{des}}$$

hence

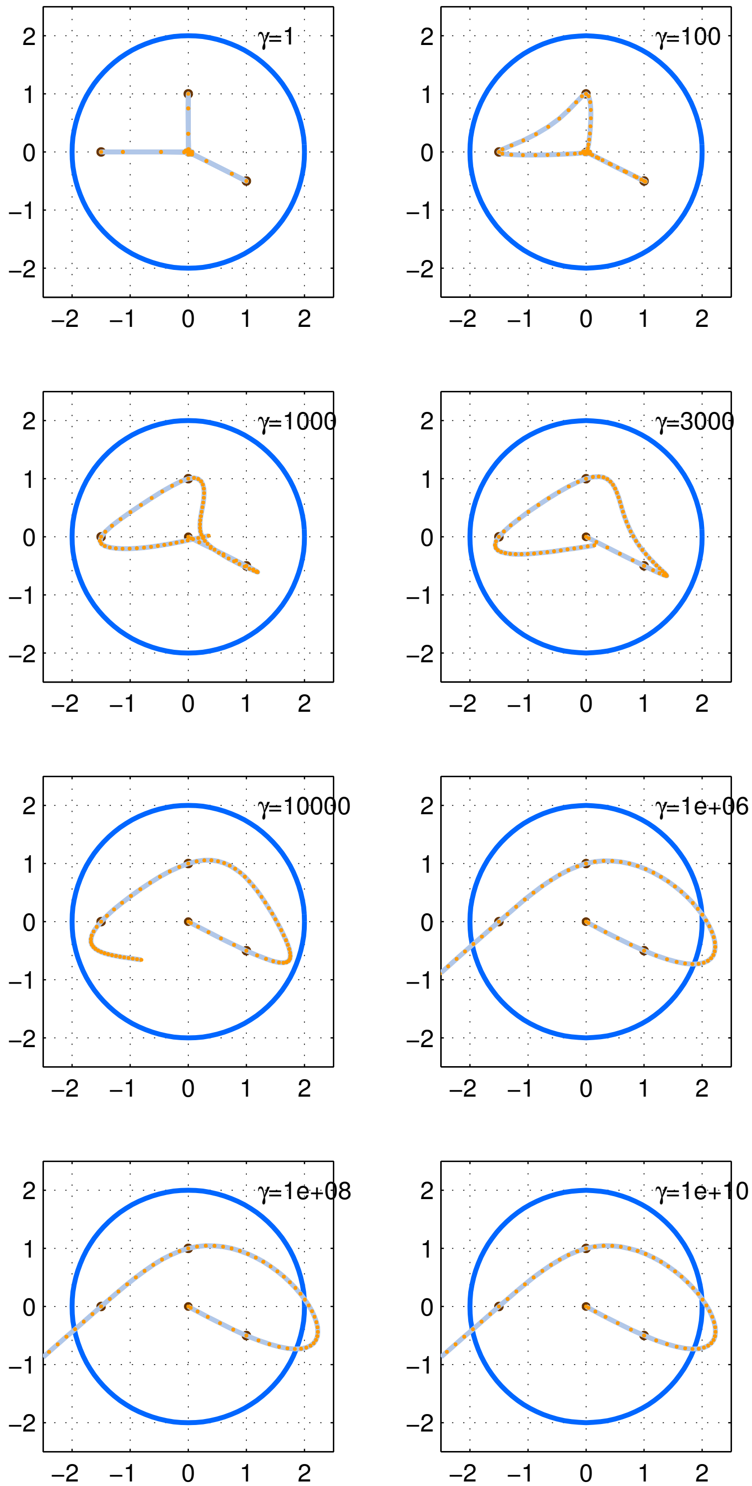
$$\lambda = -2(A\Sigma^{-1}A^T)^{-1}y_{\mathrm{des}}$$

and

$$u_{\mathrm{opt}} = \Sigma^{-1}A^T(A\Sigma^{-1}A^T)^{-1}y_{\mathrm{des}}$$

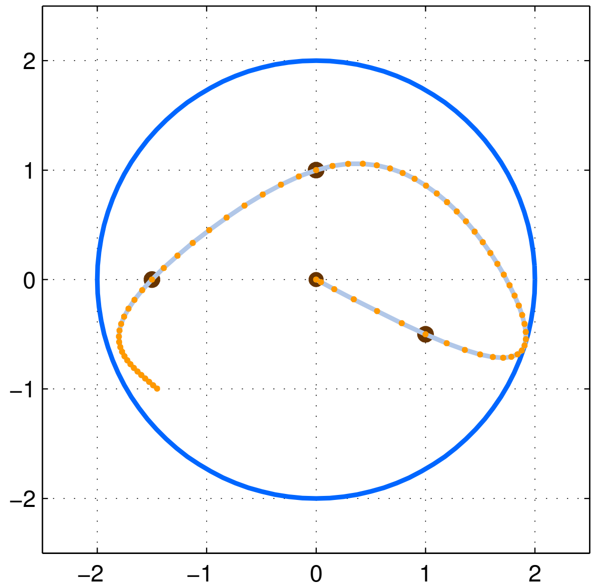
as desired.

g) The trajectory for a range of γ values is shown below. (Actually these are clearer on separate plots)



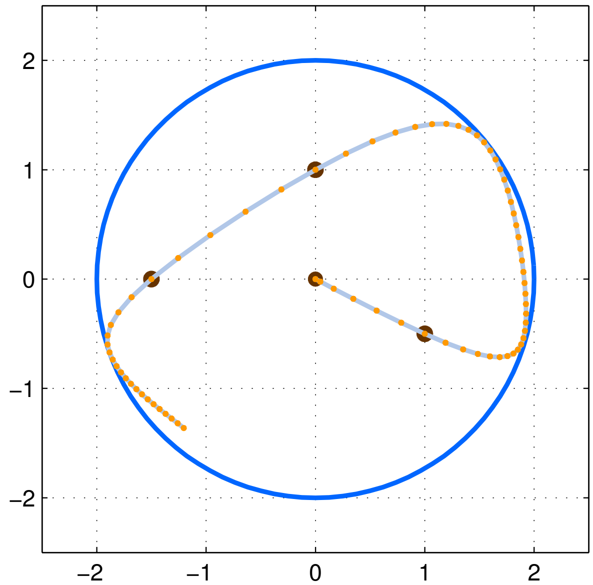
We can see the trade-off clearly; decreasing γ causes the hovercraft to try very hard to stay close to the origin. Also notice the asymmetry caused by the different times at which the hovercraft must be at the waypoints.

h) A good choice of gamma is about 1.7×10^4. Here the trajectory just remains within range, as shown below.

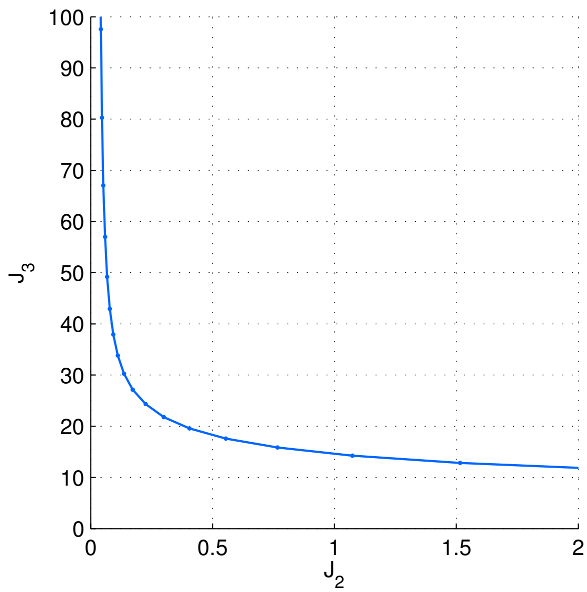


This is not the smallest-norm u that keeps the hovercraft within range and drives the hovercraft through the waypoints, because we are minimizing the *sum* of the squares of $\|q(t)\|$, rather than constraining each $\|q(t)\|$ independently. You can see this in the plot, since in the final stretch the hovercraft is expending extra effort to stay well within range, and this excessive input could be reduced.

In fact, one can compute the exact optimal, but this is not required and not covered in this course; (an approximation of) it is below.



i) The trade-off is below.



Notice that the vertical asymptote occurs when $J_2 \approx 0.03$; this is the minimum-norm of u which drives the hovercraft through the desired trajectory, as seen in part (b).

4. The smoothest input that takes the state to zero. We consider the discrete-time linear dynamical system $x(t+1) = Ax(t) + Bu(t)$, with

$$A = \begin{bmatrix} 1.0 & 0.5 & 0.25 \\ 0.25 & 0 & 1.0 \\ 1.0 & -0.5 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 1.0 \\ 0.1 \\ 0.5 \end{bmatrix}, \qquad x(0) = \begin{bmatrix} 25 \\ 0 \\ -25 \end{bmatrix}.$$

The goal is to choose an input sequence $u(0), u(1), \ldots, u(19)$ that yields $x(20) = 0$. Among the input sequences that yield $x(20) = 0$, we want the one that is *smoothest*, *i.e.*, that minimizes

$$J_{\text{smooth}} = \left(\frac{1}{20} \sum_{t=0}^{19} (u(t) - u(t-1))^2 \right)^{1/2},$$

where we take $u(-1) = 0$ in this formula. Explain how to solve this problem. Plot the smoothest input u_{smooth}, and give the associated value of J_{smooth}.

Solution. We first express $x(20)$ in terms of $u(0), \ldots, u(19)$ and $x(0)$. Using the linear recursion $x(t+1) = Ax(t) + Bu(t)$, we get $x(1) = Ax(0) + Bu(0)$,

$$\begin{aligned} x(2) &= Ax(1) + Bu(1) \\ &= A\left(Ax(0) + Bu(0)\right) + Bu(1) \\ &= A^2 x(0) + ABu(0) + Bu(1) \end{aligned}$$

Continuing in this way we get

$$x(20) = A^{20} x(0) + A^{19} B u(0) + \cdots + ABu(18) + Bu(19).$$

In matrix form, and substituting $x(20) = 0$, we have

$$\begin{bmatrix} A^{19}B & \cdots & AB & B \end{bmatrix} \begin{bmatrix} u(0) \\ \vdots \\ u(18) \\ u(19) \end{bmatrix} = -A^{20} x(0). \tag{1}$$

Thus, we have an underdetermined set of linear equations, with 3 equations and 20 variables. We know how to find the smallest (or least-norm) solution of this set of equations, but we're asked here to find the *smoothest* solution, so we have to do a little more work. We define the difference between two consecutive inputs as $\delta(t) = u(t) - u(t-1)$, with $\delta(0) = u(0)$ (consistent with $u(-1) = 0$). These are sometimes called *input increments* or *input differences*. We can express the inputs in terms of these increments as

$$u(t) = \delta(0) + \cdots + \delta(t),$$

which can be expressed in matrix form as $u = F\delta$, where $u = (u(0), \ldots, u(19))$, $\delta = (\delta(0), \ldots, \delta(19))$, and

$$F = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & & & \ddots & & \\ 1 & 1 & 1 & \cdots & 1 & 0 \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}.$$

Now, we substitute $u = F\delta$ into (1) to obtain

$$\begin{bmatrix} A^{19}B & \cdots & AB & B \end{bmatrix} F\delta = -A^{20} x(0).$$

Defining G as $G = \begin{bmatrix} A^{19}B & \cdots & AB & B \end{bmatrix} F$, we can express this as

$$G\delta = -A^{20}x(0).$$

The original smoothness measure can be expressed in terms of δ as

$$\begin{aligned} J_{\text{smooth}} &= \left(\frac{1}{20} \sum_{t=0}^{19} (u(t) - u(t-1))^2 \right)^{1/2} \\ &= \left(\frac{1}{20} \sum_{t=0}^{19} \delta(t)^2 \right)^{1/2} = \frac{1}{\sqrt{20}} \|\delta\|. \end{aligned}$$

Thus, we have reduced our problem to one we know how to solve: find the least norm vector δ_{ln} that satisfies $G\delta = -A^{20}x(0)$. The solution is given by

$$\delta_{\text{ln}} = G^{\mathsf{T}}(GG^{\mathsf{T}})^{-1}(-A^{20}x(0)).$$

Then the smoothest input is given by

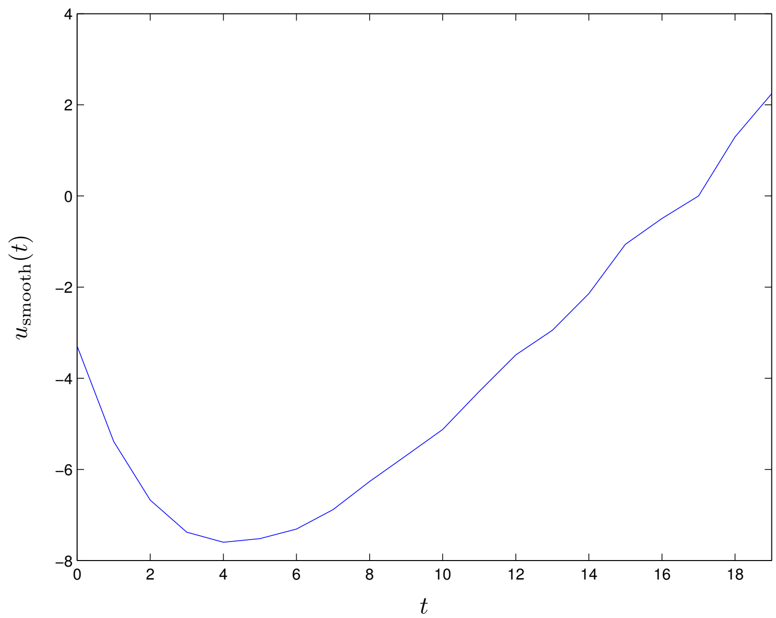
$$u_{\text{smooth}} = F\delta_{\text{ln}} = -FG^{\mathsf{T}}(GG^{\mathsf{T}})^{-1}A^{20}x(0).$$

Its RMS value is $J_{\text{smooth}} = 1/\sqrt{20}\|\delta_{\text{ln}}\|$. Below is the matlab code used to compute the smoothest input.

```
A  = [1 .5 .25; .25 0 1; 1 -0.5 0];
B  = [1; .1; .5];
x0 = [25; 0; -25];
N = 20;
Ck = [];
for k = 0:N-1
Ck = [A^k*B Ck];
end
F = tril(ones(N,N)); % lower triagonal of ones
del_smooth = pinv(Ck*F)*(-A^N*x0);
u_smooth = F*del_smooth
plot([0:N-1],u_smooth); xlabel('time'); ylabel('usmooth');
Jsmooth = 1/sqrt(N)*norm(del_smooth)
xN = A^N*x0 + Ct*u_smooth
>> xN =
1.0e-10 *
-0.0546
-0.2910
0.0909
```

The plot of the smoothest input is given below. Its RMS smoothness measure is $J_{\text{smooth}} =$

1.1246.



It is interesting to note some of the alternative methods people used to solve this problem. One method worked as follows: we have three equality constraints, *i.e.*, the constraint that the three components of $x(20)$ be zero. We can use these to eliminate three of our variables, say, $u(17)$, $u(18)$, and $u(19)$. Thus, we have an *unconstrained* problem, with 17 variables, $\tilde{u} = (u(0), \ldots, u(16))$. We can choose any values we like for these, and then $u(17)$, $u(18)$, and $u(19)$ are determined (to ensure $x(20) = 0$). Finally, we express the objective, the smoothness measure, as the norm of a linear function of $\tilde{u}$ minus a constant vector. Then we apply least-squares. (That's the idea — the details and equations are not too pretty!) This method is correct, *i.e.*, it gets the exact solution, so it got full credit. Some people formulated the the problem as a mixed quadratic problem with equality constraints, and derived the general solution of such problems. This also works. The most popular alternative approach involved a regularization method. In this method, the input u is considered unconstrained. We identify two objectives, J_{smooth} and J_{miss}, where J_{miss} gives the final state error:

$$J_{\mathrm{miss}} = \|x(20)\|.$$

We then form a regularized problem: we minimize

$$J_{\mathrm{smooth}}^2 + \lambda J_{\mathrm{miss}}^2$$

where $\lambda > 0$ is a parameter. Now we let λ get really big, which drives the final state error to be really small. This method works, at least in the limit. Even though the solution with a large value of λ is numerically very close to the exact solution, we did deduct some points for this. The reason is simple: this method is only approximate, or involves a limit. The other methods described above, in contrast, are exact.

5. **Optimal dynamic purchasing.** You are to complete a large order to buy a certain number, B, of shares in some company. You are to do this over T time periods. (Depending on the

circumstances, a single time period could be between tens of milliseconds and minutes.) We will let b_t denote the number of shares bought in time period t, for $t = 1, \ldots, T$, so we have $b_1 + \cdots + b_T = B$. (The quantities $B, b_1, \ldots, b_T$ can all be any real number; $b_t < 0$, for example, means we *sold* shares in the period t. We also don't require b_t to be integers.) We let p_t denote the price per share in period t, so the total cost of purchasing the B shares is $C = p_1 b_1 + \cdots + p_T b_T$.

The amounts we purchase are large enough to have a noticeable effect on the price of the shares. The prices change according to the following equations:

$$p_1 = \bar{p} + \alpha b_1, \qquad p_t = \theta p_{t-1} + (1-\theta)\bar{p} + \alpha b_t, \quad t = 2, \ldots, T.$$

Here $\bar{p}$ is the base price of the shares and α and θ are parameters that determine how purchases affect the prices. The parameter α, which is positive, tells us how much the price goes up in the current period when we buy one share. The parameter θ, which lies between 0 and 1, measures the *memory*: If $\theta = 0$ the share price has no memory, and the purchase made in period t only affects the price in that period; if θ is 0.5 (say), the effect a purchase has on the price decays by a factor of two between periods. If $\theta = 1$, the price has perfect memory and the price change will persist for all future periods.

If purchases didn't increase the price, the cost of purchasing the shares would always be $\bar{p}B$. The difference between the total cost and this cost, $C - \bar{p}B$, is called the *transaction cost*.

Find the purchase quantities $b_1, \ldots, b_T$ that minimize the transaction cost $C - \bar{p}B$, for the particular problem instance with

$$B = 10000, \quad T = 10, \quad \bar{p} = 10, \quad \theta = 0.8, \quad \alpha = 0.00015.$$

Give the optimal transaction cost. Also give the transaction cost if all the shares were purchased in the first period, and the transaction cost if the purchases were evenly spread over the periods (*i.e.*, if 1000 shares were purchased in each period). Compare these three quantities.

You must explain your method clearly, using any concepts from this class, such as least-squares, pseudo-inverses, eigenvalues, singular values, etc. If your method requires that some rank or other conditions to hold, say so. You must also check, in your matlab code, that these conditions are satisfied for the given problem instance.

Solution. We first derive a compact expression for p, the vector of prices, in terms of b, the vector of purchase amounts. By iterating the price process, we get

$$\begin{aligned}
p_1 &= \bar{p} + \alpha b_1 \\
p_2 &= \bar{p} + \alpha b_2 + \alpha\theta b_1 \\
p_3 &= \bar{p} + \alpha b_3 + \alpha\theta b_2 + \alpha\theta^2 b_1 \\
&\vdots \\
p_T &= \bar{p} + \alpha b_T + \alpha\theta b_{T-1} + \cdots + \alpha\theta^{T-1} b_1.
\end{aligned}$$

We write this as $p = \bar{p}\mathbf{1} + Ab$, where $A \in \mathbb{R}^{T\times T}$ is the (lower triangular Toeplitz) matrix with

$$A_{ij} = \begin{cases} \alpha\theta^{i-j} & i \geq j \\ 0 & \text{otherwise.} \end{cases}$$

The cost is

$$C = b^\mathsf{T} p = b^\mathsf{T}(\bar{p}\mathbf{1} + Ab) = \bar{p}B + b^\mathsf{T} Ab,$$

since $b^\mathsf{T}\mathbf{1} = B$. The first term, $\bar{p}B$, is just the total cost if the purchases did not increase the share price. The second term, $b^\mathsf{T} Ab$, is the transaction cost. So we see now that the transaction cost is a quadratic form in b. The matrix A is not symmetric, which can lead to trouble, so we'll write the transaction cost as $(1/2)b^\mathsf{T}(A + A^\mathsf{T})b$.

For our problem instance, we can check that A is indeed positive definite. Intuitively, this must be the case; otherwise C can be made arbitrarily negative (*i.e.*, we can make an arbitrary profit).

To find the optimal purchase quantities b, we must solve the following optimization problem:

$$\begin{array}{ll} \text{minimize} & (1/2)b^\mathsf{T}(A + A^\mathsf{T})b \\ \text{subject to} & \mathbf{1}^\mathsf{T} b = B \end{array}$$

with variable b. This not exactly a problem we've solved before, but we can solve it using methods from the notes. (We'll also see below how to convert it to a problem we have solved before.)

The Lagrangian is

$$L(b, \lambda) = (1/2)b^\mathsf{T}(A + A^\mathsf{T})b + \lambda\left(\mathbf{1}^\mathsf{T} b - B\right),$$

and the optimality conditions are

$$\nabla_b L(b, \lambda) = (A + A^\mathsf{T})b + \lambda\mathbf{1} = 0, \quad \nabla_\lambda L(b, \lambda) = \mathbf{1}^\mathsf{T} b - B = 0.$$

This can be written as

$$\begin{bmatrix} A + A^\mathsf{T} & \mathbf{1} \\ \mathbf{1}^\mathsf{T} & 0 \end{bmatrix} \begin{bmatrix} b \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ B \end{bmatrix},$$

and so

$$\begin{bmatrix} b \\ \lambda \end{bmatrix} = \begin{bmatrix} A + A^\mathsf{T} & \mathbf{1} \\ \mathbf{1}^\mathsf{T} & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ B \end{bmatrix},$$

assuming that the block matrix is invertible. (It is, for our problem instance.)

Alternatively, we can solve the optimality conditions by block elimination. First we consider $\nabla_b L(b, \lambda) = 0$, which gives $b = -\lambda(A + A^\mathsf{T})^{-1}\mathbf{1}$. The Lagrange multiplier λ is chosen to satisfy $\nabla_\lambda L(b, \lambda) = \mathbf{1}^\mathsf{T} b - B = 0$. This gives

$$\mathbf{1}^\mathsf{T} b = -\lambda\mathbf{1}^\mathsf{T}(A + A^\mathsf{T})^{-1}\mathbf{1} = B,$$

so $\lambda = -B/\mathbf{1}^\mathsf{T}(A + A^\mathsf{T})^{-1}\mathbf{1}$, and we get

$$b = \frac{B}{\mathbf{1}^\mathsf{T}(A + A^\mathsf{T})^{-1}\mathbf{1}}(A + A^\mathsf{T})^{-1}\mathbf{1}.$$

The associated transaction cost is

$$\begin{aligned} (1/2)b^\mathsf{T}(A + A^\mathsf{T})b &= \frac{B^2}{2\left(\mathbf{1}^\mathsf{T}(A + A^\mathsf{T})^{-1}\mathbf{1}\right)^2}\mathbf{1}^\mathsf{T}(A + A^\mathsf{T})^{-1}(A + A^\mathsf{T})(A + A^\mathsf{T})^{-1}\mathbf{1} \\ &= \frac{B^2}{2\left(\mathbf{1}^\mathsf{T}(A + A^\mathsf{T})^{-1}\mathbf{1}\right)}. \end{aligned}$$

Another way of solving the optimization problem is to rearrange it into a general norm minimization problem with equality constraints, and refer directly to the solution given in lecture slides 8-13 to 8-15. To do this, we note that $A + A^\mathsf{T}$ is positive definite, so we can find a symmetric matrix $F \in \mathbb{R}^{T\times T}$ for which $F^2 = A + A^\mathsf{T}$. (Indeed, we can take $F = (A + A^\mathsf{T})^{1/2}$.) This gives the optimization problem

$$\begin{array}{ll} \text{minimize} & (1/2)\|Fb\|^2 \\ \text{subject to} & \mathbf{1}^\mathsf{T} b = B. \end{array}$$

The solution then follows from lecture 8, and is, of course, exactly the same as the one given above.

The following matlab code solves the problem.

```
% problem instance
B = 10000; T = 10; pbar = 10; theta = 0.8; alpha = 0.00015;
% generate A matrix
A = zeros(T,T);
for i = 1:T for j = 1:i A(i,j) = alpha*theta^(i-j); end; end;
% check that A is positive definite
min(eig(A+A'))
% nominal cost (evenly spread purchases)
cnom = ((B/T)^2)*ones(T,1)'*A*ones(T,1);
% one period cost (all shares purchased in the first period)
conep = B^2*A(1,1);
% optimal cost
blam = [A+A',ones(T,1);ones(T,1)',0]\[zeros(T,1);B];
bopt = blam(1:T);
copt = bopt'*A*bopt;
```

For our problem instance, the transaction cost incurred if all the shares are purchased in any single period (including the first) is 15000. If the purchases are evenly spread (1000 per period), we incur a transaction cost of 4822.12. If we employ the optimal strategy, we have a transaction cost of 4688.35.

6. **Tax policies.** In this problem we explore a dynamic model of an economy, including the effects of government taxes and spending, which we assume (for simplicity) takes place at the beginning of each year. Let $x(t) \in \mathbb{R}^n$ represent the pre-tax economic activity at the beginning of year t, across n sectors, with $x(t)_i$ being the pre-tax activity level in sector i. We let $\tilde{x}(t) \in \mathbb{R}^n$ denote the post-tax economic activity, across n sectors, at the beginning of year t. We will assume that all entries of $x(0)$ are positive, which will imply that all entries of $x(t)$ and $\tilde{x}(t)$ are positive, for all $t \geq 0$.

The pre- and post-tax activity levels are related as follows. The government taxes the sector activities at rates given by $r \in \mathbb{R}^n$, with r_i the tax rate for sector i. These rates all satisfy $0 \leq r_i < 1$. The total government revenue is then $R(t) = r^\mathsf{T} x(t)$. This total revenue is then spent in the sectors proportionally, with $s \in \mathbb{R}^n$ giving the spending proportions in the sectors. These spending proportions satisfy $s_i \geq 0$ and $\sum_{i=1}^n s_i = 1$; the spending in sector i

is $s_i R(t)$. The post-tax economic activity in sector i, which accounts for the government taxes and spending, is then given by

$$\tilde{x}(t)_i = x(t)_i - r_i x(t)_i + s_i R(t), \quad i = 1, \ldots, n, \quad t = 0, 1, \ldots.$$

Economic activity propagates from year to year as $x(t+1) = E\tilde{x}(t)$, where $E \in \mathbb{R}^{n\times n}$ is the input-output matrix of the economy. You can assume that all entries of E are positive.

We let $S(t) = \sum_{i=1}^{n} x(t)_i$ denote the total economic activity in year t, and we let

$$G = \lim_{t\to\infty} \frac{S(t+1)}{S(t)}$$

denote the (asymptotic) growth rate (assuming it exceeds one) of the economy.

a) Explain why the growth rate does not depend on $x(0)$ (unless it exactly satisfies a single linear equation, which we rule out as essentially impossible). Express the growth rate G in terms of the problem data r, s, and E, using ideas from the course. You may assume that a matrix that arises in your analysis is diagonalizable and has a single dominant eigenvalue, *i.e.*, an eigenvalue λ_1 that satisfies $|\lambda_1| > |\lambda_i|$ for $i = 2, \ldots, n$. (These assumptions aren't actually needed—they're just to simplify the problem for you.)

b) Consider the problem instance with data

$$E = \begin{bmatrix} 0.3 & 0.4 & 0.1 & 0.6 \\ 0.2 & 0.3 & 0.7 & 0.2 \\ 0.1 & 0.2 & 0.2 & 0.1 \\ 0.4 & 0.2 & 0.3 & 0.2 \end{bmatrix}, \quad r = \begin{bmatrix} 0.45 \\ 0.25 \\ 0.1 \\ 0.1 \end{bmatrix}, \quad s = \begin{bmatrix} 0.15 \\ 0.3 \\ 0.4 \\ 0.15 \end{bmatrix}.$$

Find the growth rate. Now find the growth rate with the tax rate set to zero, *i.e.*, $r = 0$ (in which case s doesn't matter). You are welcome (even, encouraged) to simulate the economic activity to double-check your answer, but we want the value using the expression found in part (a).

Solution. Tracing through the equations, we get $x(t+1) = Ax(t)$, where

$$A = E(I - \mathbf{diag}(r) + sr^{\mathsf{T}}).$$

Alternatively, we can express A by its entries as

$$A_{ij} = E_{ij}(1 - r_j) + (Es)_i r_j, \quad i, j = 1, \ldots, n.$$

It follows that $x(t) = A^t x(0)$.

Using our assumption that A is diagonalizable, we have

$$x(t) = \sum_{i=1}^{n} \lambda_i^t v_i w_i^{\mathsf{T}} x(0),$$

where λ_i are the eigenvalues of A, v_i the associated eigenvectors, and w_i the associated left eigenvectors (suitably normalized). We sort the eigenvalues so λ_1 is dominant: $|\lambda_1| > |\lambda_i|$ for $i = 2, \ldots, n$. Then assuming that $w_1^{\mathsf{T}} x(0) \neq 0$, we can approximate $x(t)$ as

$$x(t) \approx \lambda_1^t v_1 (w_1^{\mathsf{T}} x(0)).$$

Now we see that λ_1 must be positive; if not, then the equation above shows $x(t)$ would eventually have negative entries, which is impossible. (The same argument shows that the entries of v_1 must be nonnegative.) Thus we have

$$S(t) \approx \lambda_1^t(\mathbf{1}^\mathsf{T} v_1)(w_1^\mathsf{T} x(0)).$$

(Note that since the entries of v_1 are nonnegative, we cannot have $\mathbf{1}^\mathsf{T} v_1 = 0$; if that were true, then $v_1 = 0$.) Thus, we have

$$S(t+1)/S(t) \approx \lambda_1.$$

So $G = \lambda_1$.

Now evaluating G for the given data, we find that $G = 1.0908$ (9.08% growth) without taxes, and $G = 1.1237$ (12.37%) with taxes. So, the taxes have actually *boosted* the asymptotic economic growth rate! Tell that to the next person who tells you that taxes are always bad (if you know any such people).

A script for computing the growth rates is given below.

```
E = [0.3 0.4 0.1 0.6;
     0.2 0.3 0.7 0.2;
     0.1 0.2 0.2 0.1;
     0.4 0.2 0.3 0.2]

r = [0.45 ; 0.25 ; 0.1; 0.1 ];
s = [0.15 ; 0.3 ; 0.4; 0.15 ];

G_no_tax = max(abs(eig(E)))
A = E*(eye(n)-diag(r)+s*r');
G_with_tax = max(abs(eig(A)))
```