EE263 Prof. S. Boyd
Oct. 26–27 or Oct. 27–28, 2007.

Midterm exam solutions

1. *Vector space multiple access (VSMA).* We consider a system of k transmitter-receiver pairs that share a common medium. The goal is for transmitter i to transmit a vector signal $x_i \in \mathbf{R}^{n_i}$ to the ith receiver, without interference from the other transmitters. All receivers have access to the same signal $y \in \mathbf{R}^m$, which includes the signals of all transmitters, according to

$$y = A_1 x_1 + \cdots + A_k x_k,$$

where $A_i \in \mathbf{R}^{m \times n_i}$. You can assume that the matrices A_i are skinny, *i.e.*, $m \geq n_i$ for $i = 1, \ldots, k$. (You can also assume that $n_i > 0$ and $A_i \neq 0$, for $i = 1, \ldots, k$.) Since the k transmitters all share the same m-dimensional vector space, we call this *vector space multiple access*. Each receiver knows the received signal y, and the matrices $A_1, \ldots, A_k$.

We say that the ith signal is *decodable* if the ith receiver can determine the value of x_i, no matter what values $x_1, \ldots, x_k$ have. Roughly speaking, this means that receiver i can process the received signal so as to perfectly recover the ith transmitted signal, while rejecting any interference from the other signals $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k$. Whether or not the ith signal is decodable depends, of course, on the matrices $A_1, \ldots, A_k$.

Here are four statements about decodability:

(a) Each of the signals $x_1, \ldots, x_k$ is decodable.

(b) The signal x_1 is decodable.

(c) The signals $x_2, \ldots, x_k$ are decodable, but x_1 isn't.

(d) The signals $x_2, \ldots, x_k$ are decodable when x_1 is 0.

For each of these statements, you are to give the exact (*i.e.*, necessary and sufficient) conditions under which the statement holds, in terms of $A_1, \ldots, A_k$ and $n_1, \ldots, n_k$. Each answer, however, must have a very specific form: it must consist of a conjunction of one or more of the following properties:

I. $\mathrm{rank}(A_1) < n_1$.

II. $\mathrm{rank}([A_2 \ \cdots \ A_k]) = n_2 + \cdots + n_k$.

III. $\mathrm{rank}([A_1 \ \cdots \ A_k]) = n_1 + \mathrm{rank}([A_2 \ \cdots \ A_k])$.

IV. $\mathrm{rank}([A_1 \ \cdots \ A_k]) = \mathrm{rank}(A_1) + \mathrm{rank}([A_2 \ \cdots \ A_k])$.

As examples, possible answers (for each statement) could be "I" or "I and II", or "I and II and IV". For some statements, there may be more than one correct answer; we will accept any correct one.

You can also give the response "My attorney has advised me not to respond to this question at this time." This response will receive partial credit.

For (just) this problem, we want only your answers. We do not want, and will not read, any further explanation or elaboration, or any other type of answers.

Solution.

Before you read the solution, please remember this: We did not ask you to prove or justify anything; we certainly did not expect you to work out complete arguments like we have below.

Let's look at what decodable means in matrix terms. First suppose that A_i has nonzero nullspace, *i.e.*, its columns are dependent. Then, even with all other x_j zero, we cannot possibly recover x_i from y. So for x_i to be decodable, we're going to need A_i to be full rank (we've already assumed it's skinny). Assuming A_i is full rank, we can decode x_i given y, when all the other x_j's are zero. Another way of saying that A_i is full rank is $\mathrm{rank}(A_i) = n_i$.

Now let's see what happens when the others start transmitting. Suppose

$$\mathcal{R}(A_i) \cap \mathcal{R}([A_1 \ \cdots \ A_{i-1} \ A_{i+1} \ \cdots \ A_k]) \neq \{0\}.$$

This means that x_i is not decodable, since there's some nonzero y that can be generated by a combination of the other transmitters, or alternatively, by transmitter i. There's no way of knowing which was the case, so y cannot be decoded, and thus x_i is not decodable.

So far we've seen that x_i is not decodable if A_i isn't full rank, or if the range intersection condition above holds. In fact the converse is true: x_i is decodable provided A_i has full rank, and

$$\mathcal{R}(A_i) \cap \mathcal{R}([A_1 \ \cdots \ A_{i-1} \ A_{i+1} \ \cdots \ A_k]) = \{0\}.$$

We'll show this now. Let's carry out a rank-revealing QR factorization of the righthand matrix, to get a skinny full rank matrix Q whose range is the range of the matrix on the right, so the condition above is $\mathcal{R}(A_i) \cap \mathcal{R}(Q) = \{0\}$. The matrix $[A_i \ Q]$ is skinny and full rank. Indeed, if this were not the case, there would be a nonzero vector (u, v) for which $A_i u + Qv = 0$. This means that $A_i u$, which is in $\mathcal{R}(A_i)$, is equal to $-Qv$, which is in $\mathcal{R}(Q)$. Thus, both are zero. But at least one of u and v is nonzero: If $u \neq 0$, then $A_i u = 0$, which contradicts our assumption that A_i is full rank; if $v \neq 0$, then $Qv = 0$, which contradicts our assumption that Q is full rank.

The next step is to find a left inverse B of $[A_i \ Q]$ (which is possible since $[A_i \ Q]$ is skinny and full rank). Let's write B as

$$B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix},$$

where B_1 has n_i rows. Then $B[A_i\ Q] = I$ implies that

$$B_1 A_i = I, \qquad B_1 Q = 0.$$

Now we're done, because, for any values of $x_1, \ldots, x_k$, we have

$$B_1 y = B_1 A_i x_i + B_1(A_1 x_1 + \cdots + A_{i-1} x_{i-1} + A_{i+1} x_{i+1} + \cdots + A_k x_k) = x_i.$$

So, we've constructed a perfect (linear) decoder for x_i. Whew.

Now we we turn to the statements.

(b) Now, taking $i = 1$, we have the exact conditions under which x_1 is decodable: $\mathrm{rank}(A_1) = n_1$ and

$$\mathcal{R}(A_i) \cap \mathcal{R}([A_1\ \cdots\ A_{i-1}\ A_{i+1}\ \cdots\ A_k]) = \{0\}.$$

This last condition is equivalent to

$$\mathrm{rank}([A_1\ \cdots\ A_k]) = \mathrm{rank}(A_1) + \mathrm{rank}([A_2\ \cdots\ A_k]).$$

Combining the two conditions, we get

$$\mathrm{rank}([A_1\ \cdots\ A_k]) = n_1 + \mathrm{rank}([A_2\ \cdots\ A_k]),$$

which is Property III. We could add Property IV as well, and state the answer as III & IV. But the answer IV alone is wrong. (Property IV, for example, holds when $A_1 = 0$, even though in this case x_1 is certainly not decodable.)

(a) The statement 'Each of the signals $x_1, \ldots, x_k$ is decodable' is equivalent to $[A_1\ \cdots\ A_k]$ being full rank, *i.e.*, having rank $n_1 + \cdots + n_k$. The same argument as above works. We construct a left inverse B of this matrix, and we chop up its rows into blocks, the first one with n_1 rows, and so on. Then we have $B_i y = x_i$ no matter what the x_i are. Conversely, if the matrix is less than full rank (or not skinny), it has a nonzero vector in its nullspace; this corresponds to two possible different x's that generate the same y.

Now let's form the answer. One choice is II & III, which together tell us that

$$\mathrm{rank}([A_1\ \cdots\ A_k]) = n_1 + \cdots + n_k,$$

which is our condition. We can also add in condition IV with no harm.

(d) Now let's look at the statement 'The signals $x_2, \ldots, x_k$ are decodable when x_1 is 0'. This is essentially the same as Statement (a), with $K - 1$ transmitters, and x_1 just removed. It is equivalent to

$$\mathrm{rank}([A_2\ \cdots\ A_k]) = n_2 + \cdots + n_k,$$

which is condition II.

(c) Now we come to the statement 'The signals $x_2, \ldots, x_k$ are decodable, but x_1 isn't'. If this is true, then $\text{rank}([A_2 \ \cdots \ A_k]) = n_2 + \cdots + n_k$ (which is the condition that $x_2, \ldots, x_k$ are decodable when $x_1 = 0$). This is condition II. Now we look at

$$\mathcal{R}(A_1) \cap \mathcal{R}([A_2 \ \cdots \ A_k]).$$

If this subspace were nonzero, then a nonzero signal from transmitter 1 produces exactly the same signal as come combination of the signals $x_2, \ldots, x_k$. This would imply that at least one of $x_2, \ldots, x_k$ is not decodable, and so it must be false. In other words, the intersection above is $\{0\}$. It follows that

$$\text{rank}([A_1 \ \cdots \ A_k]) = \text{rank}(A_1) + \text{rank}([A_2 \ \cdots \ A_k]),$$

which is condition IV. By our previous conclusion, we must have $\text{rank}(A_1) < n_1$; otherwise, all signals would be decodable. Thus, condition I holds.

We can interpret what we have so far. It means that x_1 and the other signals cannot interfere (if they did, we could not decode the other signals). But this means that the other signals do not interfere with x_1. Since x_1 is not decodable, this must be because of self-interference; in other words, two values of x_1 produce the same y. This means $\text{rank}(A_1) < n_1$.

So far, we've argued that if the statement above holds, then conditions I, II, and IV must hold. In fact, the converse is also true: if these conditions hold, then the statement holds. From I, we see that x_1 is not decodable. To show that $x_2, \ldots, x_k$ are decodable, we can use the same type of construction used above. We find a matrix Q, with independent columns with the same range as A_1. (It will have $r = \text{rank}(A_1)$ columns.) We then argue that the matrix $[A_2 \ \cdots \ A_k \ Q]$ is skinny and full rank, using II and IV. We find a left inverse B, and partition it into rows of height $n_2, \ldots, n_k, r$. Then we have, for any x, $B_i y = x_i$, for $i = 2, \ldots, k$.

2. *Signal reconstruction for a bandlimited signal.* In this problem we refer to *signals*, which are just vectors, with index interpreted as (discrete) time. It is common to write the index for a signal as an argument, rather than as a subscript; for example, if $y \in \mathbf{R}^N$ is a signal, we use $y(t)$ to denote y_t, with $t \in \{1, 2, \ldots, N\}$. Another notation you'll sometimes see in signal processing texts is $y[t]$ for y_t.

   The *discrete cosine transformation* (DCT) of the signal $y \in \mathbf{R}^N$ is another signal, typically denoted using the corresponding upper case symbol $Y \in \mathbf{R}^N$. It is defined as

$$Y(k) = \sum_{t=1}^{N} y(t)w(k) \cos \frac{\pi(2t-1)(k-1)}{2N}, \quad k = 1, \ldots, N,$$

   where $w(k)$ are weights, with

$$w(k) = \begin{cases} \sqrt{1/N}, & k = 1, \\ \sqrt{2/N}, & k = 2, \ldots, N. \end{cases}$$

   The number $Y(k)$ is referred to as the *kth DCT coefficient of* y. The *DCT bandwidth* of the signal y is the smallest K for which $Y(K) \neq 0$, and $Y(k) = 0$ for $k = K+1, \ldots, N$. When $K < N$, the signal is called *DCT bandlimited*. (The term is typically used to refer to the case when the DCT bandwidth, K, is significantly smaller than N.)

   A signal y can be reconstructed from its DCT Y, via the *inverse DCT transform*, with

$$y(t) = \sum_{k=1}^{N} Y(k)w(k) \cos \frac{\pi(2t-1)(k-1)}{2N}, \quad t = 1, \ldots, N,$$

   where $w(k)$ are the same weights as those used above in the DCT.

   Now for the problem. You are given noise-corrupted values of a DCT bandlimited signal y, at some (integer) times $t_1, \ldots, t_M$, where $1 \leq t_1 < t_2 < \cdots < t_M \leq N$:

$$y_i^{\text{samp}} = y(t_i) + v_i, \quad i = 1, \ldots, M.$$

   Here, v_i are small noises or errors. You don't know v, but you do know that its RMS value is approximately σ, a known constant. (In signal processing, y^{samp} would be called a non-uniformly sampled, noise corrupted version of y.)

   Your job is to

   - Determine the smallest DCT bandwidth (*i.e.*, the smallest K) that y could have.
   - Find an estimate of y, $\hat{y}$, which has this bandwidth.

   Your estimate $\hat{y}$ must be consistent with the sample measurements y^{samp}. While it need not match exactly (you were told there was a small amount of noise in y^{samp}), you should ensure that the vector of differences,

$$(y_1^{\text{samp}} - \hat{y}(t_1^{\text{samp}}), \ldots, y_M^{\text{samp}} - \hat{y}(t_M^{\text{samp}})),$$

   has a small RMS value, on the order of σ (and certainly no more than 3σ).

(a) Clearly describe how to solve this problem. You can use any concepts we have used, to date, in EE263. *You cannot use (and do not need) any concepts from outside the class. This includes the Fourier transform and other signal processing methods you might know.*

(b) Carry out your method on the data in `bandlimit.m`. Running this script will define `N`, `ysamp`, `M`, `tsamp`, and `sigma`. It will also plot the sampled signal.

Give K, your estimate of the DCT bandwidth of y. Show $\hat{y}$ on the same plot as the original sampled signal. (We have added the command to do this in `bandlimit.m`, but commented it out.)

Also, give us $\hat{y}(129)$, to four significant figures.

You might find the Matlab functions `dct` and `idct` useful; `dct(eye(N))` and `idct(eye(N))` will return matrices whose columns are the DCT, and inverse DCT transforms, respectively, of the unit vectors. Note, however, that you can solve the problem without using these functions.

Solution. We start by defining a selector matrix S, which is the identity matrix with appropriate rows taken out so that $y^{\text{samp}} = Sy + v$. Here $v = [v_1 \ \cdots \ v_n]^T$. Next we let $A \in \mathbf{R}^{n\times n}$ be the appropriate inverse DCT transform matrix, so that $y = AY$, where Y is the vector of DCT coefficients of the signal y. Thus we have

$$y^{\text{samp}} = SAY + v.$$

Here the only thing we know about v is that it has an RMS value of approximately σ. We also don't know Y. We have the prior information that $Y(k) = 0$ for $k > K$, but, sadly, we don't know K.

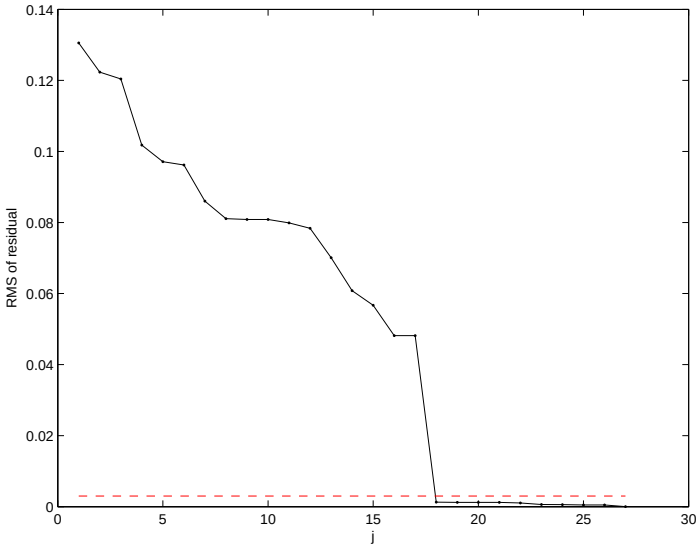
The key to solving this problem is to rephrase the statements above as: y^{samp} is a noise corrupted linear combination of the first K columns of SA. Now we have a way to solve the problem. For $j = 1, 2, \ldots$ we carry out a least-squares fit of y^{samp} as a linear combination of the first j columns of SA, and examine the residual vector. The RMS value of this residual vector decreases as we increase j. The first time the RMS value of the residual is on the order of σ, we declare that value of j as our estimate of the DCT bandwidth K. Once we have guessed K, we use $\hat{Y}$, the least-squares estimate of Y (which has zero entries for $k > K$) to construct our estimate $\hat{y} = A\hat{Y}$.

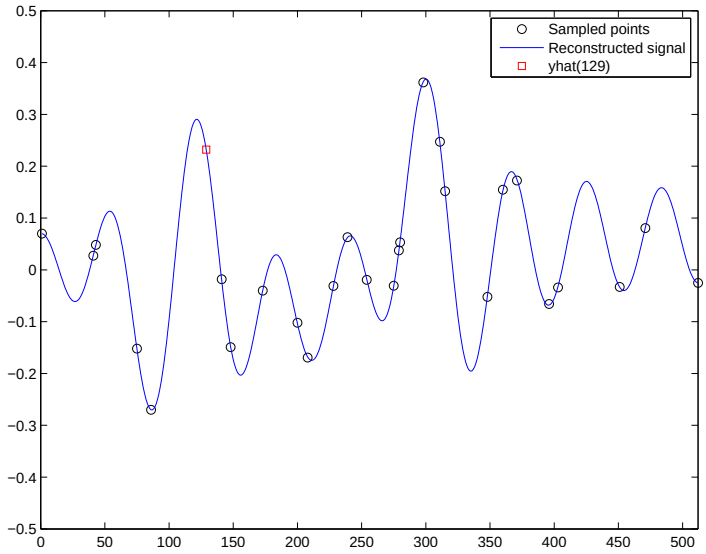
The following Matlab code plots the RMS value of the residual, for $j = 1, \ldots, N$. We also plot σ as a dotted horizontal line, for comparison.

```
A = idct(eye(N));
S = eye(N);
S = S(tsamp, :);
SA = S*A; % so that ysamp = S*y+v = SA*Y+v.

ks = 1:T;
```

```
err = zeros(T, 1);
for k = ks
Ynz = SA(:, 1:k) \ ysamp;
yhat = A*[Ynz; zeros(N-k, 1)];
vhat = ysamp - S*yhat;
err(k) = sqrt(mean(square(vhat)));
end

figure(1);
cla();
plot(ks, err, 'k.-');
hold on;
plot(ks, sigma*ones(size(ks)), 'r--');
```

The plot is shown below. From the plot, we see a sharp drop in residual, to a value consistent with σ, when $j = 18$. Thus we guess $K = 18$. The resulting estimate $\hat{y}$ is also shown below. The estimate has $\hat{\sigma} = 0.0013$ and $\hat{y}(129) = 0.2320$.





The code used for the final part is shown below.

```
% Select the point with 18 coefficients.
K = 18;
Ynz = SA(:, 1:K) \ ysamp;
yhat = A*[Ynz; zeros(N-K, 1)];

figure(2);
cla;
plot(tsamp, ysamp, 'ko');
hold on;
plot(yhat, 'b')
plot(129, yhat(129), 'rs')
legend('Sampled points', 'Reconstructed signal', 'yhat(129)')
axis([0 512 -0.5 0.5])
print -depsc2 figures/lbw2.eps

sqrt(mean(square(S*yhat - ysamp)))
yhat(129)
```

3. *Fitting a model for hourly temperature.* You are given a set of temperature measurements (in degrees C), $y_t \in \mathbf{R}$, $t = 1, \ldots, N$, 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 $a \in \mathbf{R}$ and $p \in \mathbf{R}^N$ 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 $a \in \mathbf{R}$ and $p \in \mathbf{R}^N$ (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 `tempdata.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 file `tempdata.m`, 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

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

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

The solution is given by

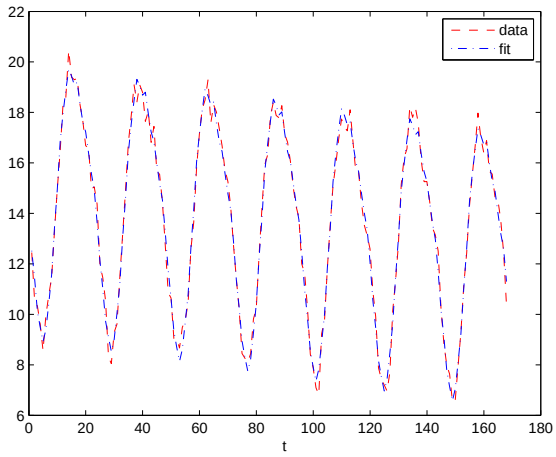
$$x = (A^T A)^{-1} A^T y.$$

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

(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.



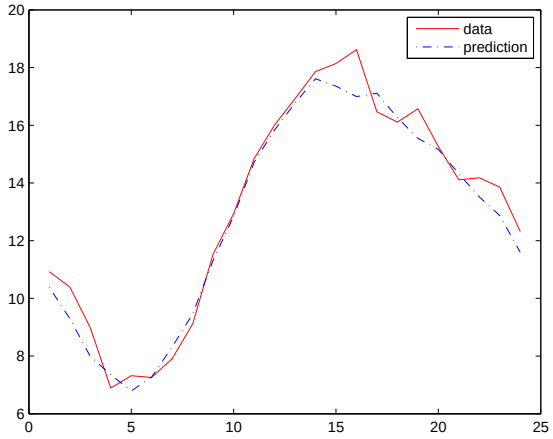
(c) In part (b) we estimated the values of a and p_i, $i = 1, \ldots, 24$, collected together as one vector $x \in \mathbf{R}^{25}$. To estimate y_{tom}, the 24-vector of tomorrow's temperatures, we use

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

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.

4. *Minimum energy roundtrip.* We consider the linear dynamical system

$$x(t+1) = Ax(t) + Bu(t), \quad x(0) = 0,$$

with $u(t) \in \mathbf{R}$ and $x(t) \in \mathbf{R}^n$. We must choose $u(0), u(1), \ldots, u(T-1)$ so that $x(T) = 0$ (*i.e.*, after T steps we are back at the zero state), and $x(t_{\text{dest}}) = x_{\text{dest}}$ (*i.e.*, at time t_{dest} the state is equal to x_{dest}). Here $x_{\text{dest}} \in \mathbf{R}^n$ is a given destination state. The time t_{dest} is *not* given; it can be any integer between 1 and $T-1$. The goal is to minimize the total input energy, defined as

$$E = \sum_{t=0}^{T-1} u(t)^2.$$

Note that you have to find t_{dest}, the time when the state hits the desired state, as well as the input trajectory $u(0), \ldots, u(T-1)$.

(a) Explain how to do this. For this problem, you may assume that $n \leq t_{\text{dest}} \leq T-n$. If you need some matrix or matrices that arise in your analysis to be full rank, you can just assume they are. But you must state this clearly.

(b) Carry out your method on the particular problem instance with data

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad T = 30, \quad x_{\text{dest}} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}.$$

Give the optimal value of t_{dest} and the associated value of E, and plot the optimal input trajectory u.

Solution. We first derive an expression for $x(t)$ in terms of $u(0), \ldots, u(T-1)$, by just iterating the basic dynamics equation:

$$x(t) = A^{t-1}Bu(0) + \cdots + ABu(t-2) + Bu(t-1).$$

Let's assume that t_{dest} is known. We have requirements on $x(t_{\text{dest}})$ and also on $x(T)$:

$$\begin{aligned} x_{\text{dest}} &= A^{t_{\text{dest}}-1}Bu(0) + \cdots + Bu(t_{\text{dest}}-1) \\ 0 &= A^{T-1}Bu(0) + \cdots + Bu(T-1). \end{aligned}$$

Let's write this as a matrix equation,

$$Gu = h,$$

where $u_i = u(i-1)$, $i = 1, \ldots, T$, and $G \in \mathbf{R}^{2n \times T}$, defined as

$$G = \begin{bmatrix} A^{t_{\text{dest}}-1}B & \cdots & B & 0 & \cdots & 0 \\ A^{T-1}B & \cdots & & & \cdots & B \end{bmatrix},$$

and

$$h = \begin{bmatrix} x_{\text{dest}} \\ 0 \end{bmatrix}.$$

Now we can restate the problem as: Find the u that satisfies $Gu = h$ and minimizes $\|u\|$. Assuming G has rank $2n$, the optimal u is

$$u = G^T(GG^T)^{-1}h.$$

The corresponding energy is

$$E = \|u\|^2 = h^T(GG^T)^{-1}h.$$

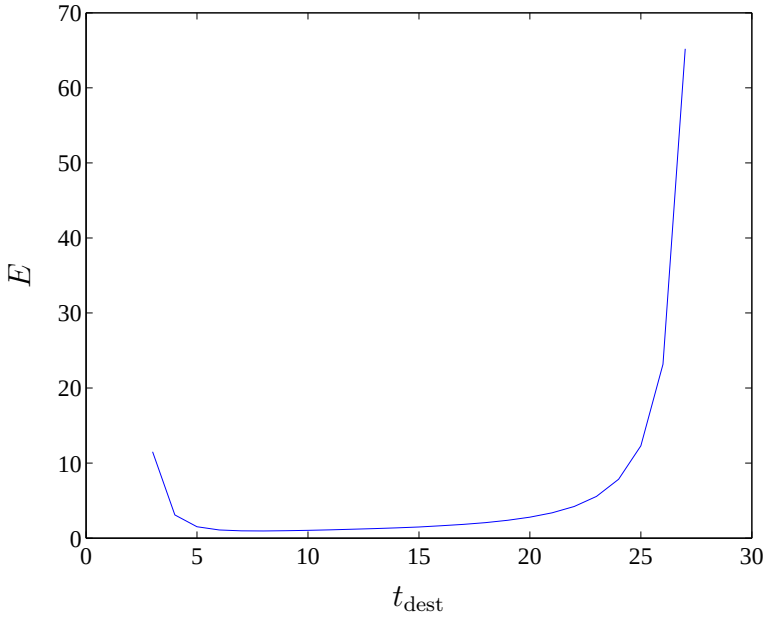
So far we have assumed that t_{dest} is fixed. (We need to know t_{dest} to form the matrix G.) There's no analytical method to find the best value of t_{dest}; so we'll just try each value, between n and $T - n$, and record the associated value of E. Whichever one has the lowest value of E is the one we want.

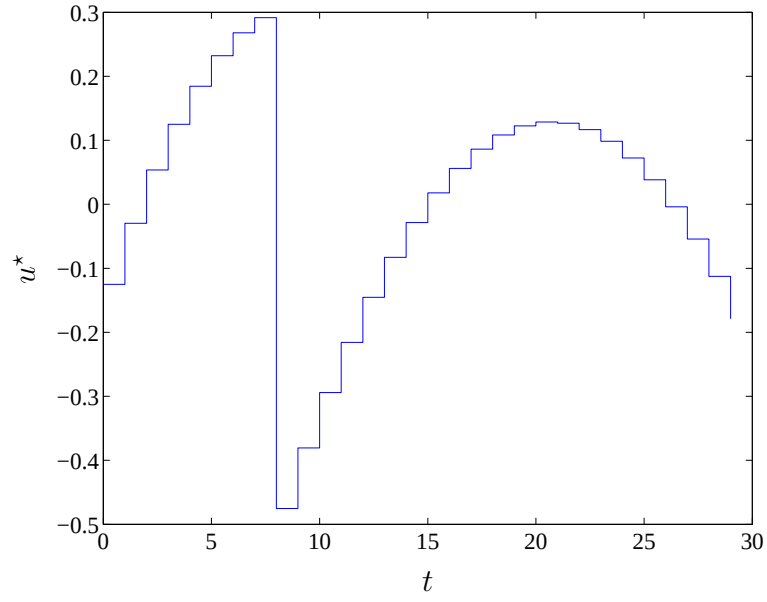
Applying the method we have just described to the problem data, we obtain an optimal $t^\star_{\text{dest}} = 8$, and $E^\star = 0.9603$. This is computed using the following matlab script.

```
% model parameters
A = [1 0 0; 1 1 0; 0 1 1];
B = [1;0;0];
n = 3; % size of A
T = 30;
xdest = [1;1;-1];

% set up matrices
h = [xdest; zeros(n,1)];
% first G for tdest = T
Gbottom = zeros(n,T);
for t = 1:T
    Gbottom(:,T-t+1) = A^(t-1)*B;
end

% calculate minimum energy for each tdest
E = [];
tdests = n:T-n; % range of tdest's to check
for tdest = tdests
    % form matrix G
    Gtop = [Gbottom(:,T-tdest+1:T), zeros(n,T-tdest)];
    G = [Gtop; Gbottom];
    E = [E, h'*inv(G*G')*h];
end
```

```
% plot the energy vs. tdest
figure(1); plot(tdests,E);
set(gca,'FontName','times','FontSize',16);
ylabel('E'); xlabel('tdest');
print('-depsc','figures/minert_e.eps');

% extract the optimal trajectory
[Estar,ind] = min(E);
tdest_star = tdests(ind);
Gtop = [Gbottom(:,T-tdest_star+1:T), zeros(n,T-tdest_star)];
G = [Gtop; Gbottom];
ustar = G'*inv(G*G')*h;

% plot the optimal trajectory
tvec = 0:1:T-1;
figure(2); stairs(tvec,ustar);
set(gca,'FontName','times','FontSize',16);
ylabel('u'); xlabel('t');
print('-depsc','figures/minert_u.eps');
```

The following are plots of E versus t_{dest}, and the optimal trajectory $u^\star(t)$.


70
60
50
40
30
20
10
0
0
5
10
15
20
25
30
E
t_{dest}


0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−0.5
0
5
10
15
20
25
30
$u^\star$
t

5. *Empirical algorithm complexity.* The runtime T of an algorithm depends on its input data, which is characterized by three key parameters: k, m, and n. (These are typically integers that give the dimensions of the problem data.) A simple and standard model that shows how T scales with k, m, and n has the form

$$\hat{T} = \alpha k^{\beta} m^{\gamma} n^{\delta},$$

where α, β, γ, $\delta \in \mathbf{R}$ are constants that characterize the approximate runtime model. If, for example, $\delta \approx 3$, we say that the algorithm has (approximately) cubic complexity in n. (In general, the exponents β, γ, and δ need not be integers, or close to integers.)

Now suppose you are given measured runtimes for N executions of the algorithm, with different sets of input data. For each data record, you are given T_i (the runtime), and the parameters k_i, m_i, and n_i. It's possible (and often occurs) that two data records have identical values of k, m, and n, but different values of T. This means the algorithm was run on two different data sets that had the same dimensions; the corresponding runtimes can be (and often are) a little different.

We wish to find values of α, β, γ, and δ for which our model (approximately) fits our measurements. We define the fitting cost as

$$J = (1/N) \sum_{i=1}^{N} \left(\log(\hat{T}_i / T_i) \right)^2,$$

where $\hat{T}_i = \alpha k_i^{\beta} m_i^{\gamma} n_i^{\delta}$ is the runtime predicted by our model, using the given parameter values. This fitting cost can be (loosely) interpreted in terms of relative or percentage fit. If $(\log(\hat{T}_i/T_i))^2 \leq \epsilon$, then $\hat{T}_i$ lies between $T_i/\exp\sqrt{\epsilon}$ and $T_i \exp\sqrt{\epsilon}$.

Your task is to find constants α, β, γ, δ that minimize J.

(a) Explain how to do this. If your method always finds the values that give the true global minimum value of J, say so. If your algorithm cannot guarantee finding the true global minimum, say so. If your method requires some matrix (or matrices) to be full rank, say so.

(b) Carry out your method on the data found in `empac_data.m`. Give the values of α, β, γ, and δ you find, and the corresponding value of J.

Solution. We can write

$$\begin{aligned} J &= \frac{1}{N} \sum_{i=1}^{N} \left(\log \frac{\hat{T}_i}{T_i} \right)^2 \\ &= \frac{1}{N} \sum_{i=1}^{N} \left(\log \alpha + \beta \log k_i + \gamma \log m_i + \delta \log n_i - \log T_i \right)^2 \\ &= \frac{1}{N} \left\| Ax - b \right\|^2, \end{aligned}$$

where

$$A = \begin{bmatrix} 1 & \log k_1 & \log m_1 & \log n_1 \\ 1 & \log k_2 & \log m_2 & \log n_2 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \log k_N & \log m_N & \log n_N \end{bmatrix}, \qquad b = \begin{bmatrix} \log(T_1) \\ \log(T_2) \\ \vdots \\ \log(T_N) \end{bmatrix},$$

and $x = \begin{bmatrix} \log\alpha & \beta & \gamma & \delta \end{bmatrix}^T$. The solution (by least squares) is $x^\star = (A^TA)^{-1}A^Tb$.

Running the following matlab script on our dataset gives $\hat{\alpha} = \exp(x_1^\star) = 9.3364 \times 10^{-15}$, $\hat{\beta} = x_2^\star = 3.1062$, $\hat{\gamma} = x_3^\star = 1.0943$, and $\hat{\delta} = x_4^\star = 2.1078$. The cost in this case is $J^\star = 0.0113$.

For comparison, the values we used to generate the data are $\alpha = 1 \times 10^{-14}$, $\beta = 3.1$, $\gamma = 1.1$, and $\delta = 2.1$.

```
empac_data;
% form the matrices
A = [ones(N,1),log(k),log(m),log(n)]; b = log(T);
% solve by least-squares
xstar = A\b;
% extract the estimated constants
alpha = exp(xstar(1)); beta = xstar(2); gamma = xstar(3); delta = xstar(4);
% compute J
Jstar = (1/N)*norm(A*xstar-b)^2;
```