

## Midterm exam solutions

1. *Element-wise nonnegative matrix and inverse.* Suppose a matrix  $A \in \mathbf{R}^{n \times n}$ , and its inverse  $B$ , have all their elements nonnegative, *i.e.*,  $A_{ij} \geq 0$ ,  $B_{ij} \geq 0$ , for  $i, j = 1, \dots, n$ . What can you say must be true of  $A$  and  $B$ ?

Please give your answer first, and then the justification. Your solution (which includes what you can say about  $A$  and  $B$ , as well as your justification) must be short; we won't read more than one page. We want the most specific description possible; no credit will be given for true but obvious statements (such as that  $A$  and  $B$  must be invertible).

**Solution.** We will show that each row and column of  $A$  and  $B$  has exactly one nonzero entry. Moreover, the sparsity patterns of  $A$  and  $B^T$  are the same; that is,  $A_{ij} \neq 0$  if and only if  $B_{ji} \neq 0$ . In fact, we have  $B_{ji} = 1/A_{ij}$ .

We can state this several other ways (and we accepted lots of ways to say this!). For example, we can say that  $A$  is a product of a nonsingular diagonal matrix and a permutation matrix. The condition above follows, since if  $A = \mathbf{diag}(d_1, \dots, d_n)P$ , where  $d_i \neq 0$ , and  $P$  is a permutation matrix, then  $A^{-1} = P^T \mathbf{diag}(1/d_1, \dots, 1/d_n)$ .

Let's show that  $A$  and  $B$  must have this structure.

First, if any row or column of  $A$  is zero,  $A$  will be rank deficient and hence noninvertible, so each row or column of  $A$  must have at least one nonzero entry.

Next, suppose that in the  $i$ th row of  $A$  we have two nonzero entries,  $A_{ip}$  and  $A_{iq}$ . Since  $AB = I$ , we have, for  $j \neq i$ ,

$$I_{ij} = 0 = \sum_{k=1}^n A_{ik}B_{kj} \geq A_{ip}B_{pj} + A_{iq}B_{qj},$$

where the inequality is by nonnegativity of all products in the sum. We must, therefore, have  $B_{pj} = B_{qj} = 0$ , for all  $j \neq i$ . Thus, the  $p$ th and  $q$ th rows of  $B$  are zero in all except (possibly) their  $i$ th columns. This means that the  $p$ th or  $q$ th row of  $B$  is a (possibly zero) multiple of the other, so  $B$  is rank deficient. This contradicts the assumption that  $B = A^{-1}$ , so the rows of  $A$  must have at most one nonzero entry.

So now we know that each row of  $A$  has exactly one nonzero entry. Different rows must have different nonzero entries; otherwise one row of  $A$  is a (possibly zero) multiple of another row of  $A$ , and  $A$  would be singular. Thus, the sparsity pattern of  $A$  is that of a permutation: each row and each column has exactly one nonzero element.

The same argument shows that  $B$  has the same form. Or, we can conclude it directly from what we know about  $A$ , because a matrix with exactly one nonzero in each row and each column has an inverse with the same property. Indeed, we can explicitly give construct  $B = A^{-1}$  as

$$B_{ij} = \begin{cases} 1/A_{ji}, & A_{ji} \neq 0 \\ 0, & \text{otherwise.} \end{cases}$$

**Alternative solution:** Let  $C = AB$ . We can write

$$C = a_1 b_1^T + a_2 b_2^T + \cdots + a_n b_n^T$$

where  $a_1, \dots, a_n$  are the columns of  $A$  and  $b_1^T, \dots, b_n^T$  denote the rows of  $B$ . We know that both  $A$  and  $B$  are invertible, so each row and column of  $A$  and  $B$  must contain at least one nonzero element. Now suppose that the  $i$ th column of  $A$ ,  $a_i$ , contains more than one nonzero entry. The  $i$ th row of  $B$ ,  $b_i^T$ , must contain at least one nonzero, and so every nonzero column of the outer product

$$a_i b_i^T = \begin{bmatrix} b_{i1} a_i & b_{i2} a_i & \cdots & b_{in} a_i \end{bmatrix}$$

must contain more than one nonzero entry. It follows that we cannot have  $C = I$  (since the outer products  $a_i b_i^T$ ,  $i = 1, \dots, n$ , are elementwise nonnegative). Furthermore, different columns of  $A$  must have different nonzero entries, otherwise one column of  $A$  is a multiple of another column, and  $A$  would be singular. So we conclude that each row and column of  $A$  must have exactly one nonzero element, and by symmetry, each row and column of  $B$  must have exactly one nonzero element.

2. *Least-squares classification.* For each of  $N$  documents we are given a feature vector  $x^{(i)} \in \mathbf{R}^n$ , and a label  $y_i \in \{-1, 1\}$ . (This is called a binary label.) Each component of the feature vector could be, for example, the number of occurrences of a certain term in the document; the label could be decided by a person working with the documents, with  $+1$  meaning the document is interesting or useful, and  $-1$  meaning the document is not (for example, spam). From this data set we construct  $w \in \mathbf{R}^n$  and  $v \in \mathbf{R}$  that minimize

$$\sum_{i=1}^N (w^T x^{(i)} + v - y_i)^2.$$

We can now use  $w$  and  $v$  to predict the label for other documents, *i.e.*, to guess whether an as-yet-unread document is interesting, by forming  $\hat{y} = \mathbf{sign}(w^T x + v)$ . For scalar  $a$ , we define  $\mathbf{sign}(a) = +1$  for  $a \geq 0$  and  $\mathbf{sign}(a) = -1$  for  $a < 0$ ; for vector arguments,  $\mathbf{sign}()$  is taken elementwise.

- (a) Explain (briefly) how to find  $w$  and  $v$ . If you need to make an assumption about the rank of a matrix, say so.

- (b) Find  $w$  and  $v$  for the data in `ls_classify_data.m`, which defines  $\mathbf{X}$ , whose columns are  $x^{(i)}$ , and  $\mathbf{y}$ . This M-file will also define a second data set, `Xtest` and `ytest`, of the same size (*i.e.*,  $n$  and  $N$ ). Use the  $w$  and  $v$  you found to make predictions about whether the documents in the test set are interesting. Give the number of correct predictions (for which  $\hat{y}_i = y_i$ ), false positives ( $\hat{y}_i = +1$  while  $y_i = -1$ ), and false negatives ( $\hat{y}_i = -1$  while  $y_i = +1$ ) for the test set. You may find the Matlab function `sign()` useful. To count false positives, for example, you can use `sum((yhat == 1) & (y == -1))`.

*Remark.* There are better methods for binary classification, which you can learn about in a modern statistics or machine learning course, or in EE364a. But least-squares classification can sometimes work well.

**Solution.**

- (a) We can write the objective as

$$\|X^T w + v \mathbf{1} - y\|^2,$$

where

$$X = \begin{bmatrix} x^{(1)} & x^{(2)} & \dots & x^{(N)} \end{bmatrix}.$$

To find  $w$  and  $v$  we solve the least squares problem

$$\text{minimize} \quad \left\| \begin{bmatrix} X^T & \mathbf{1} \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} - y \right\|^2.$$

At this point in the course, we'd need to assume that the matrix above is skinny and full rank.

- (b) The following Matlab code solves for  $w$  and  $v$ , and implements our classifier on the test data set.

```
ls_classify_data;

% solve for w and v
wv = [X',ones(N,1)]\y;
w = wv(1:n); v = wv(n+1);

% yhat for test set
yhat = sign(Xtest'*w+v);
ncorrect = sum(yhat==ytest) % correct predictions
nfp = sum((yhat==1)&(ytest==-1)) % false positives
nfn = sum((yhat==-1)&(ytest==1)) % false positives
```

For the 100 test examples we achieve 92 correct predictions, 6 false positives and 2 false negatives.

3. *Minimum time control.* We consider a discrete-time linear dynamical system

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

with  $x(t) \in \mathbf{R}^n$ ,  $u(t) \in \mathbf{R}^m$ .

- (a) You are given  $A$ ,  $B$ , and  $x(0) = x_{\text{init}}$ . Explain how to find an input sequence  $u(0), u(1), \dots, u(N-1)$ , so that  $x(N) = 0$ , with  $N$  is as small as possible. Your answer can involve any of the concepts used in the course so far, *e.g.*, range, rank, nullspace, least-squares,  $QR$  factorization, etc.
- (b) Apply the method described in part (a) to the specific problem instance with data

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \quad x_{\text{init}} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

(where  $n = 4$  and  $m = 1$ ). You must give us the (minimum) value of  $N$ , and a sequence of inputs  $u(0), \dots, u(N-1)$  that results in  $x(N) = 0$ .

**Solution.**

- (a) We can express  $x(N)$  in terms of  $u(0), \dots, u(N-1)$ , by iterating the basic dynamics equation:

$$x(N) = A^{N-1}Bu(0) + A^{N-2}Bu(1) + \dots + ABu(N-2) + Bu(N-1) + A^N x_{\text{init}}.$$

We can write the condition that  $x(N) = 0$  as

$$\begin{bmatrix} A^{N-1}B & A^{N-2}B & \dots & AB & B \end{bmatrix} \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N-1) \end{bmatrix} = -A^N x_{\text{init}}.$$

This holds for some input sequence if and only if

$$-A^N x_{\text{init}} \in \mathcal{R} \left( \begin{bmatrix} A^{N-1}B & A^{N-2}B & \dots & AB & B \end{bmatrix} \right).$$

The smallest possible  $N$  is found by checking this condition for  $N = 1, 2, \dots$ , until it is satisfied.

- (b) The following Matlab code solves the problem.

```
% dynamics matrices
n = 4; m = 1;
A = [0,0,1,1;0,1,1,1;1,0,1,0;1,1,1,0];
B = [0,1,0,1]';
```

```

xinit = [1,1,1,1]';

% check the condition
for N = 1:10
    CN = [];
    for i = 1:N CN = [A^(i-1)*B,CN]; end;
    U = -pinv(CN)*(A^N)*xinit;
    if (norm(CN*U+(A^N)*xinit) <= 1e-6) Nmin = N; break; end;
end

```

The minimum value of  $N$  is  $N = 3$ . A sequence of inputs that achieves  $x(3) = 0$  is

$$u(0) = -9, \quad u(1) = -2, \quad u(2) = 12.$$

In fact, this is the unique input that yields  $x(3) = 0$ .

4. *Interference cancelling equalizers.* Two vector signals  $x \in \mathbf{R}^p$  and  $y \in \mathbf{R}^q$  are to be transmitted to two receivers. The transmitter broadcasts the signal  $z = Ax + By \in \mathbf{R}^n$  to each receiver. (The matrices  $A$  and  $B$  are called the coding matrices, and are known.) Receiver 1 forms an estimate of the signal  $x$  using the linear decoder  $\hat{x} = Fz$ ; receiver 2 forms an estimate of the signal  $y$  using the linear decoder  $\hat{y} = Gz$ . (The matrices  $F \in \mathbf{R}^{p \times n}$  and  $G \in \mathbf{R}^{q \times n}$  are called the decoding matrices.)

The goal is to find  $F$  and  $G$  so that  $\hat{x} = x$  and  $\hat{y} = y$ , no matter what values  $x$  and  $y$  take. This means that both decoders are perfect; each reconstructs the exact desired signal, while completely rejecting the other (undesired) signal. For this reason we call decoding matrices with this property *perfect*.

- (a) When is it possible to find perfect decoding matrices  $F$  and  $G$ ? (The conditions, of course, depend on  $A$  and  $B$ .) Your answer can involve any of the concepts we've seen so far in EE263.
- (b) Suppose that  $A$  and  $B$  satisfy the conditions in part (a). How would you find perfect decoding matrices that, among all perfect decoding matrices, minimize

$$\sum_{i=1}^p \sum_{j=1}^n F_{ij}^2 + \sum_{i=1}^q \sum_{j=1}^n G_{ij}^2.$$

We call such decoding matrices minimum norm perfect decoding matrices.

- (c) Find minimum norm perfect decoding matrices for the data (*i.e.*,  $A$  and  $B$ ) given in the M-file `mn_perf_dec_data.m`.

### Solution.

- (a) Perfect decoding matrices should reconstruct the original signal, *i.e.*, we must have

$$\begin{bmatrix} F \\ G \end{bmatrix} z = \begin{bmatrix} F \\ G \end{bmatrix} \begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$$

for any  $x$  and  $y$ . This occurs if and only if

$$\begin{bmatrix} F \\ G \end{bmatrix} \begin{bmatrix} A & B \end{bmatrix} = I,$$

i.e.,  $\begin{bmatrix} F \\ G \end{bmatrix}$  is a left inverse of  $\begin{bmatrix} A & B \end{bmatrix}$ . This is possible if and only if the matrix  $\begin{bmatrix} A & B \end{bmatrix}$  is full rank and skinny (or square).

The same condition can also be expressed as  $A$  and  $B$  are (each) full rank and skinny, and  $\mathcal{R}(A) \cap \mathcal{R}(B) = \{0\}$ .

(b) From Lecture 5-14, the smallest left inverse is the pseudo inverse,

$$\begin{bmatrix} F \\ G \end{bmatrix} = (\begin{bmatrix} A & B \end{bmatrix}^T \begin{bmatrix} A & B \end{bmatrix})^{-1} \begin{bmatrix} A & B \end{bmatrix}^T = \begin{bmatrix} A^T A & A^T B \\ B^T A & B^T B \end{bmatrix}^{-1} \begin{bmatrix} A^T \\ B^T \end{bmatrix}.$$

(c) The following Matlab script implements the solution described in part (b).

```
mn_perf_dec_data;
H=pinv([A B]); % pseudo-inverse
F=H(1:p,:);
G=H(p+1:p+q,:);
```

This gives the following minimum norm perfect decoding matrices:

$$F^T = \begin{bmatrix} -0.0224 & 0.2757 & 0.2424 \\ -0.2823 & 0.1702 & -0.0063 \\ 0.2434 & -0.0187 & -0.0889 \\ 0.0771 & 0.1582 & -0.0777 \\ -0.1110 & 0.0074 & -0.0041 \\ 0.0092 & 0.3403 & -0.0271 \\ 0.2749 & -0.0246 & 0.1410 \\ -0.0143 & 0.1828 & 0.3284 \end{bmatrix}, \quad G^T = \begin{bmatrix} -0.0801 & 0.1379 \\ 0.1107 & 0.0428 \\ 0.2473 & 0.4632 \\ -0.1611 & 0.2022 \\ -0.1498 & 0.0792 \\ 0.0326 & -0.0214 \\ -0.0366 & 0.1092 \\ 0.0510 & -0.0416 \end{bmatrix}.$$

(We give the transposes so they fit on the page.)

5. *Piecewise affine fitting.* In this problem we refer to vectors in  $\mathbf{R}^N$  as signals. We say that a signal  $z$  is *piecewise-affine* (PWA), with kink points  $i_1, \dots, i_{K+1}$ , which are integers satisfying  $i_1 = 1 < i_2 < \dots < i_{K+1} = N + 1$ , if

$$z_j = \alpha_k j + \beta_k, \quad \text{for } i_k \leq j < i_{k+1},$$

for  $k = 1, \dots, K$ . Thus, the signal value is an affine function of the index  $j$  (which we might interpret as time in this problem) over the (integer) intervals

$$1, \dots, i_2 - 1; \quad i_2, \dots, i_3 - 1; \quad \dots \quad i_K, \dots, N.$$

We call  $\alpha_k$  and  $\beta_k$  the slope and offset, respectively, in the  $k$ th interval. (It is very common to refer to such a signal as piecewise-linear, since ‘linear’ is sometimes used to mean ‘affine’.)

We can also add a *continuity requirement*,

$$\alpha_k i_{k+1} + \beta_k = \alpha_{k+1} i_{k+1} + \beta_{k+1}, \quad k = 1, \dots, K - 1.$$

This means that if each piecewise affine segment were extrapolated to the next index, the value would agree with the starting value for the next segment. When a PWA signal satisfies this condition, we say that it is continuous. (Of course, it doesn’t make sense to refer to a discrete signal as continuous; this is just special notation for PWA signals that refers to the condition above.)

Finally, we get to the problem.

- (a) You are given a signal  $y \in \mathbf{R}^N$ , and some kink points  $i_1, \dots, i_{K+1}$ . How would one find the best PWA approximation  $\hat{y}^{\text{pwa}}$  of  $y$ , with approximation error measured in the RMS sense,

$$\left( \frac{1}{N} \sum_{j=1}^N (\hat{y}_j^{\text{pwa}} - y_j)^2 \right)^{1/2}.$$

- (b) Repeat part (a), but this time, you are to find the continuous PWA approximation  $\hat{y}^{\text{pwac}}$  that minimizes the RMS deviation from  $y$ .
- (c) Carry out your methods from parts (a) and (b) on the data given in `pwa_data.m`. Running this data file will define  $y$  and the kink points. The data file also includes a commented out code template for plotting your results. Using this template, plot the original signal along with the PWA and continuous PWA approximations. Give us the RMS approximation error in both cases.

### Solution.

- (a) Let’s define basis functions as:

$$f_k(j) = \begin{cases} j - i_k & i_k \leq j < i_{k+1} \\ 0 & \text{otherwise,} \end{cases} \quad g_k(j) = \begin{cases} 1 & i_k \leq j < i_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

for  $k = 1, \dots, K$ . Then our PWA fitting at  $j$  can be written as:

$$\hat{y}_j^{\text{pwa}} = \sum_{k=1}^K (a_k f_k(j) + b_k g_k(j)).$$

This relation can be rewritten for  $j = 1, \dots, N$  as:

$$\hat{y}^{\text{pwa}} = \underbrace{\begin{bmatrix} f_1(1) & \cdots & f_K(1) & g_1(1) & \cdots & g_K(1) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ f_1(N) & \cdots & f_K(N) & g_1(N) & \cdots & g_K(N) \end{bmatrix}}_C \begin{bmatrix} a_1 \\ \vdots \\ a_K \\ b_1 \\ \vdots \\ b_K \end{bmatrix}.$$

Since  $C$  is skinny and full rank, we can find  $a_k$  and  $b_k$  minimizing  $\|\hat{y}^{\text{pwa}} - y\|^2$  by least-squares, *i.e.*,  $C^\dagger y$ .

- (b) Since  $\hat{y}^{\text{pwac}}$  is continuous at each kink point,  $b_1, \dots, b_K$  cannot be determined independently. We have  $b_{k+1} = a_k(i_{k+1} - i_k) + b_k$  which means  $b_{k+1} = \sum_{m=1}^k a_m(i_{m+1} - i_m) + b_1$  for  $k = 1, \dots, K - 1$ . In a matrix form:

$$\begin{bmatrix} b_2 \\ \vdots \\ b_K \end{bmatrix} = \underbrace{\begin{bmatrix} i_2 - i_1 & 0 & 0 & \cdots & 0 & 0 & 1 \\ i_2 - i_1 & i_3 - i_2 & 0 & \cdots & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ i_2 - i_1 & i_3 - i_2 & i_4 - i_3 & \cdots & i_K - i_{K-1} & 0 & 1 \end{bmatrix}}_D \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{K-1} \\ a_K \\ b_1 \end{bmatrix}.$$

Combined with the result of the previous problem,  $\hat{y}^{\text{pwac}}$  can be expressed as:

$$\hat{y}^{\text{pwac}} = \underbrace{C \begin{bmatrix} I_{K+1} \\ D \end{bmatrix}}_H \begin{bmatrix} a_1 \\ \vdots \\ a_K \\ b_1 \end{bmatrix}.$$

Here  $H$  is skinny and full rank, and thus we can determine  $a_k$  and  $b_1$  by least-squares, *i.e.*,  $H^\dagger y$ .

**Alternative solution:** Instead of using basis functions, we can determine the continuous piecewise affine fitting by selecting estimation values at the kink points. Since the described piecewise affine fitting results in piecewise affine functions linked at kink points, we should have

$$\begin{aligned} y_j^{\text{pwac}} &= \frac{y_{i_{k+1}}^{\text{pwac}} - y_{i_k}^{\text{pwac}}}{i_{k+1} - i_k} (j - i_k) + y_{i_k}^{\text{pwac}} \\ &= \frac{j - i_{k+1}}{i_k - i_{k+1}} y_{i_k}^{\text{pwac}} + \frac{j - i_k}{i_{k+1} - i_k} y_{i_{k+1}}^{\text{pwac}}, \quad \text{for } i_k \leq j < i_{k+1}, \end{aligned}$$

for  $k = 1, \dots, K$ .

Now, let's define  $x \in \mathbf{R}^{K+1}$  to be the estimation at kink points, *i.e.*,

$$x = \begin{bmatrix} y_{i_1}^{\text{pwac}} \\ \vdots \\ y_{i_{K+1}}^{\text{pwac}} \end{bmatrix}.$$

With this  $x$ , we can rewrite  $y^{\text{pwac}}$  as  $y^{\text{pwac}} = Ax$ , where

$$A_{jl} = \begin{cases} \frac{j-i_{k+1}}{i_k-i_{k+1}}, & \text{for } l = k \\ \frac{j-i_k}{i_{k+1}-i_k}, & \text{for } l = k + 1 \\ 0, & \text{otherwise} \end{cases}$$

for  $i_k \leq j < i_{k+1}$ ,  $k = 1, \dots, K$ , and  $l = 1, \dots, K + 1$ .

Finally, we have  $\|y^{\text{pwac}} - y\|_2 = \|Ax - y\|_2$  for given  $A \in \mathbf{R}^{N \times K}$  as above, and thus,  $y^{\text{pwac}} = Ax_{\text{ls}} = A(A^T A)^{-1} A^T y$ . Of course,  $x_{\text{ls}}$  minimizes  $\|Ax - y\|_2$ .

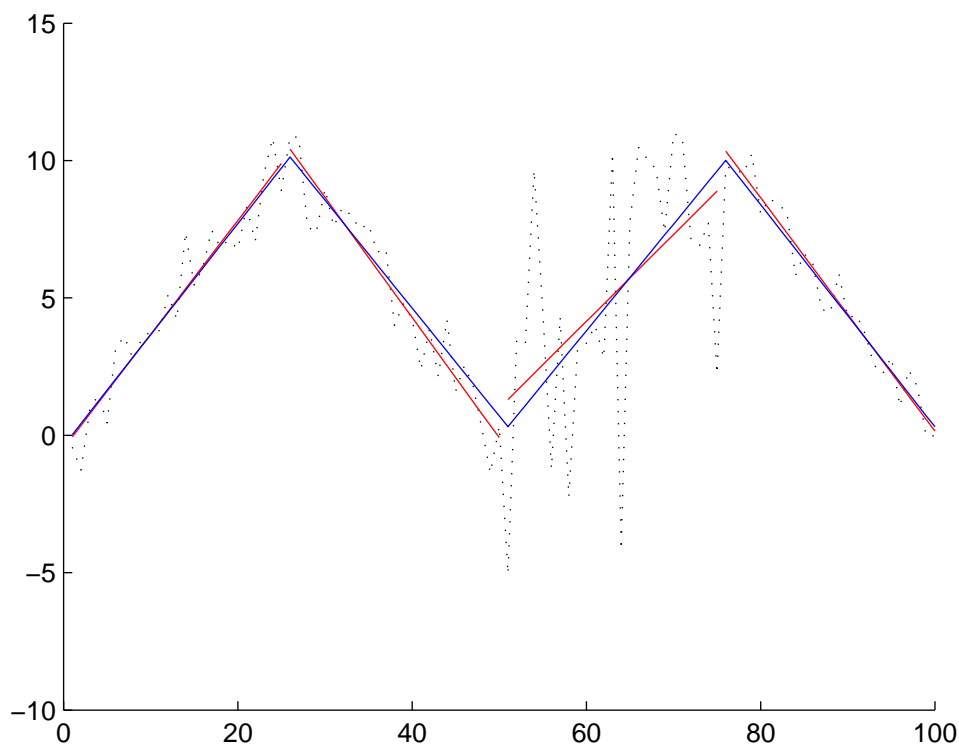
(c) This matlab code solves the problem.

```
pwa_data;

% non-continuous case
C=zeros(N,2*K);
for k=1:K
    for j = i(k):i(k+1)-1
        C(j,k)=j-i(k); %f_k(j)
        C(j,k+K)=1; %g_k(j)
    end
end
y_pwa=C*pinv(C)*y;

% continuous case
D=zeros(K-1,K+1);
for k=1:K-1
    for l=1:k
        D(k,l)=i(l+1)-i(l);
    end
    D(k,K+1)=1;
end
H=C*[eye(K+1); D];
y_pwac=H*pinv(H)*y;
```

The RMS errors of PWA and continuous PWA are 2.1290 and 2.1576, respectively. The figure below shows the given signal, piecewise affine fitting, and continuous piecewise affine fitting.



### Alternative solution 2:

(a) We can write the approximation error as

$$\left( \frac{1}{N} \sum_{k=1}^K \|A^{(k)} x^{(k)} - y^{(k)}\|^2 \right)^{1/2}$$

where

$$A^{(k)} = \begin{bmatrix} i_k & 1 \\ \vdots & \vdots \\ i_{k+1} - 1 & 1 \end{bmatrix}, \quad y^{(k)} = \begin{bmatrix} y_{i_k} \\ \vdots \\ y_{i_{k+1}-1} \end{bmatrix}, \quad x^{(k)} = \begin{bmatrix} \alpha_k \\ \beta_k \end{bmatrix},$$

for  $k = 1, \dots, K$ . We see that the objective is a sum of  $K$  expressions, and the  $k$ th expression depends only on  $x^{(k)}$ . This means that to minimize the objective, we can minimize each expression separately. Since the matrices  $A^{(1)}, \dots, A^{(K)}$  are skinny full rank, the solution is

$$\begin{bmatrix} \alpha_k \\ \beta_k \end{bmatrix} = (A^{(k)T} A^{(k)})^{-1} A^{(k)T} y^{(k)}, \quad k = 1, \dots, K.$$

(b) We can write the continuity requirement as

$$Cx = 0,$$

where

$$C = \begin{bmatrix} i_2 & 1 & -i_2 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & i_3 & 1 & -i_3 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & i_K & 1 & -i_K & 1 \end{bmatrix},$$

and  $x = [\alpha_1 \ \beta_1 \ \cdots \ \alpha_K \ \beta_K]^T$  is the vector of slope and offsets. The objective from part (a) can be expressed as

$$\left( \frac{1}{N} \|Ax - y\|^2 \right)^{1/2}$$

where

$$A = \begin{bmatrix} A^{(1)} & 0 & \cdots & 0 \\ 0 & A^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & A^{(K)} \end{bmatrix}.$$

Thus we can write the continuous PWA fitting problem as

$$\begin{aligned} & \text{minimize} && \|Ax - y\|^2 \\ & \text{subject to} && Cx = 0. \end{aligned}$$

We recognize this as a general norm minimization problem. Since  $A$  is skinny and full rank, the solution is

$$x = (A^T A)^{-1} \left( A^T y - C^T (C(A^T A)^{-1} C^T)^{-1} C(A^T A)^{-1} A^T y \right).$$

(c) The following matlab code solves this problem

```
pwa_data

% non-continuous case
y_pwa = zeros(N,1);
for k = 1:K
    inds = [i(k):1:i(k+1)-1]';
    Ak = [inds,ones(length(inds),1)]; bk = y(inds);
    xk = pinv(Ak)*bk; alphak = xk(1); betak = xk(2);
    y_pwa(inds) = alphak*inds+betak;
end
J_pwa = (1/sqrt(N))*norm(y-y_pwa);
```

```

% continuous case
y_pwac = zeros(N,1); A = zeros(N,2*K);
for k = 1:K
    inds = [i(k):1:i(k+1)-1]';
    Ak = [inds,ones(length(inds),1)];
    A(inds,(k-1)*2+1:k*2) = Ak;
end
B = zeros(K-1,4);
B(:,1) = i(2:K); B(:,2) = ones(K-1,1);
B(:,3) = -i(2:K); B(:,4) = -ones(K-1,1);
C = zeros(K-1,2*K);
for k = 1:K-1
    C(k,(k-1)*2+1:(k-1)*2+4) = B(k,:);
end
x = pinv(A'*A)*(A'*y-C'*pinv(C*pinv(A'*A)*C'))*(C*pinv(A'*A)*A'*y);
y_pwac = A*x;
J_pwac = (1/sqrt(N))*norm(y-y_pwac);

```

6. *Robust input design.* We are given a system, which we know follows  $y = Ax$ , with  $A \in \mathbf{R}^{m \times n}$ . Our goal is to choose the input  $x \in \mathbf{R}^n$  so that  $y \approx y^{\text{des}}$ , where  $y^{\text{des}} \in \mathbf{R}^m$  is a given target outcome. We'll assume that  $m \leq n$ , *i.e.*, we have more degrees of freedom in our choice of input than specifications for the outcome. If we knew  $A$ , we could use standard EE263 methods to choose  $x$ . The catch here, though, is that we don't know  $A$  exactly; it varies a bit, say, day to day. But we do have some possible values of  $A$ ,

$$A^{(1)}, \dots, A^{(K)},$$

which might, for example, be obtained by measurements of  $A$  taken on different days. We now define  $y^{(i)} = A^{(i)}x$ , for  $i = 1, \dots, K$ . Our goal is to choose  $x$  so that  $y^{(i)} \approx y^{\text{des}}$ , for  $i = 1, \dots, K$ .

We will consider two different methods to choose  $x$ .

- *Least norm method.* Define  $\bar{A} = (1/K) \sum_{i=1}^K A^{(i)}$ . Choose  $x^{\text{ln}}$  to be the least-norm solution of  $\bar{A}x = y^{\text{des}}$ . (You can assume that  $\bar{A}$  is full rank.)
- *Mean-square error minimization method.* Choose  $x^{\text{mmse}}$  to minimize the mean-square error

$$\frac{1}{K} \sum_{i=1}^K \|y^{(i)} - y^{\text{des}}\|^2.$$

- (a) Give formulas for  $x^{\text{ln}}$  and  $x^{\text{mmse}}$ , in terms of  $y^{\text{des}}$  and  $A^{(1)}, \dots, A^{(K)}$ . You can make any needed rank assumptions about matrices that come up, but please state them explicitly.

- (b) Find  $x^{\text{ln}}$  and  $x^{\text{mmse}}$  for the problem with data given in `rob_inp_des_data.m`. Running this M-file will define `ydes` and the matrices  $A^{(i)}$  (given as a 3 dimensional array; for example, `A(:, :, 13)` is  $A^{(13)}$ ). Also included in the data file (commented out) is code to produce scatter plots of your results. Write down the values of  $x^{\text{ln}}$  and  $x^{\text{mmse}}$  you found. Produce and submit scatter plots of  $y^{(i)}$  for  $x^{\text{ln}}$  and  $x^{\text{mmse}}$ . Use the code we provided as a template for your plots.

### Solution

- (a) The least norm solution is given by

$$\begin{aligned} x^{\text{ln}} &= \bar{A}^T (\bar{A} \bar{A}^T)^{-1} y^{\text{des}} \\ &= \left( \frac{1}{K} \sum_{i=1}^K A^{(i)T} \right) \left\{ \left( \frac{1}{K} \sum_{i=1}^K A^{(i)} \right) \left( \frac{1}{K} \sum_{i=1}^K A^{(i)T} \right) \right\}^{-1} y^{\text{des}}. \end{aligned}$$

Our assumption that  $\bar{A}$  is full rank guarantees that the inverses above make sense. To get a formula for  $x^{\text{mmse}}$ , let's write  $\tilde{y} = \tilde{A}x$ , where

$$\tilde{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(K)} \end{bmatrix}, \quad \tilde{A} = \begin{bmatrix} A^{(1)} \\ \vdots \\ A^{(K)} \end{bmatrix}.$$

The mean-square error is

$$\frac{1}{K} \sum_{i=1}^K \|y^{(i)} - y^{\text{des}}\|^2 = \frac{1}{K} \|\tilde{y} - \tilde{y}^{\text{des}}\|^2 = \frac{1}{K} \|\tilde{A}x - \tilde{y}^{\text{des}}\|^2,$$

where

$$\tilde{y}^{\text{des}} = \begin{bmatrix} y^{\text{des}} \\ \vdots \\ y^{\text{des}} \end{bmatrix}.$$

Now we've got a standard least-squares problem, with solution

$$\begin{aligned} x^{\text{mmse}} &= (\tilde{A}^T \tilde{A})^{-1} \tilde{A}^T \tilde{y}^{\text{des}} \\ &= \left( \sum_{i=1}^K A^{(i)T} A^{(i)} \right)^{-1} \left( \sum_{i=1}^K A^{(i)T} y^{\text{des}} \right). \end{aligned}$$

Here we are making the assumption that  $\tilde{A}$  is skinny and full rank, *i.e.*, has rank  $n$ .

Note that the formulas for  $x^{\text{ln}}$  and  $x^{\text{mmse}}$  look rather similar, but are not the same.

- (b) The following matlab code solves part (b).

```

clear all;
clc

rob_inp_des_data;

A_bar = (1/K)*sum(A,3); % average over 3rd index

% least norm solution
x_ln = pinv(A_bar)*y_des;

% mmse method
A_til= []; y_til_des=[];

% loop to concatenate A^(i) matrices and y_des
for i=1:K
    A_til = [A_til; A(:, :, i)];
    y_til_des = [y_til_des; y_des];
end

% least-squares solution
x_mmse = A_til \ y_til_des;

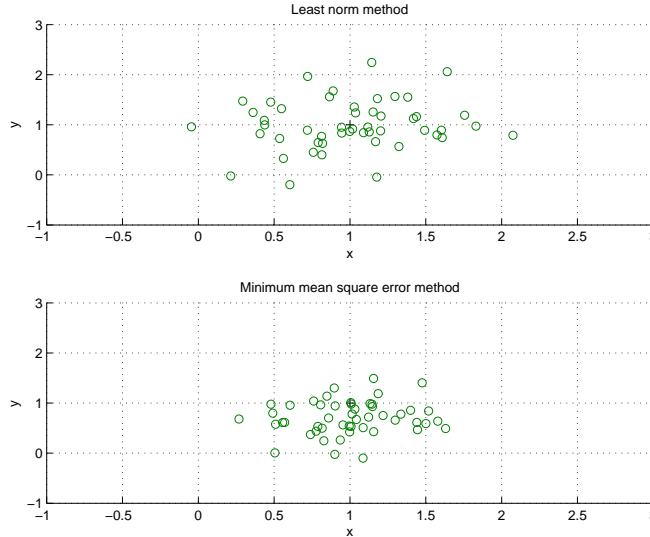
% compute y estimates
y_test_ln = A_til*x_ln; y_ln = reshape(y_test_ln,m,K);
y_test_mmse = A_til*x_mmse; y_mmse = reshape(y_test_mmse,m,K);

% plots
figure
subplot(211)
hold on; scatter(y_des(1),y_des(2),'k+');
scatter(y_ln(1,:),y_ln(2,:));hold off
xlabel('x');ylabel('y');title('Least norm method')
axis([-1 3 -1 3]);grid

subplot(212)
hold on; scatter(y_des(1),y_des(2),'k+');
scatter(y_mmse(1,:),y_mmse(2,:));hold off;
xlabel('x');ylabel('y');title('Minimum mean square error method')
axis([-1 3 -1 3]);grid
print -depsc rob_inp_des.eps

```

The  $y$ 's estimated from both methods are presented in the following figure.



7. *Unbiased estimation with deadlines.* We consider a standard measurement set up, with  $y = Ax + v$ , where  $y \in \mathbf{R}^m$  is the vector of measurements,  $v \in \mathbf{R}^m$  is the vector of measurement noises,  $x \in \mathbf{R}^n$  is the vector of parameters to be estimated, and  $A \in \mathbf{R}^{m \times n}$  characterizes the measurement system. In this problem, you should think of the index for  $y$  as denoting a time period, and you should imagine the measurements (*i.e.*, the components of  $y$ ) as arriving sequentially. In the first time period,  $y_1$  becomes available, in the next time period  $y_2$  becomes available, and so on, so that all  $m$  measurements are available in the  $m$ th time period.

You are to design a linear estimator, given by a matrix  $B \in \mathbf{R}^{n \times m}$ , with the estimate of  $x$  given by  $\hat{x} = By$ . We require that the estimator be unbiased, *i.e.*, that  $\hat{x} = x$  when  $v = 0$ .

In addition, we have *deadline constraints*, which we now explain. We require that  $\hat{x}_i$  can be computed after  $k_i$  time periods, *i.e.*, we require that  $\hat{x}_i$  must be a function of  $y_1, \dots, y_{k_i}$  only. We say that  $k_i$  is the *deadline* for computing  $\hat{x}_i$ , our estimate of the  $i$ th parameter to be estimated. You are given increasing deadlines,

$$0 < k_1 < k_2 < \dots < k_n = m.$$

Thus,  $\hat{x}_1$  may only be computed from  $y_1, \dots, y_{k_1}$ , while  $\hat{x}_n$  may be computed from all of the measurements  $y_1, \dots, y_m$ .

The data in this problem are the measurement matrix  $A$  and the deadlines  $k_1, \dots, k_n$ .

- (a) How would you determine whether or not an unbiased linear estimator, which respects the given deadlines, exists? Your answer does not have to be a single condition, such as ‘ $A$  is skinny and full rank’; it can involve a sequence of tests.
- (b) Assume that it is possible to find an unbiased linear estimator that respects the deadlines. Explain how to find the smallest such estimator matrix, *i.e.*, the  $B$

that minimizes

$$J = \sum_{i=1}^n \sum_{j=1}^m B_{ij}^2.$$

If your method requires some matrix or matrices to be full rank, you can just assume they are, but you must state this clearly.

- (c) Carry out the method described in part (b) on the data found in the M-file `unbd1_data.m`. Compare the value of  $J$  found for your estimator with the value of  $J$  for  $B = A^\dagger$ . The increase in  $J$  can be thought of as the cost of imposing the deadlines, in terms of the size of estimator matrix.

### Solution.

- (a) If  $B$  is an unbiased estimator, then we must have  $\hat{x} = x$  when  $v = 0$ . This implies that  $x = By = BAx$ , for all  $x \in \mathbf{R}^n$ , so  $BA = I$  (*i.e.*,  $B$  is a left inverse of  $A$ ).

The deadline constraints imply that  $\hat{x}_i$  can only depend on the first  $k_i$  measurements,  $y_1, \dots, y_{k_i}$ . Thus we require  $B_{ij} = 0$  for  $j > k_i$ . Let  $b_1^T, \dots, b_n^T$  denote the rows of  $B$ . We can write

$$b_i^T = \begin{bmatrix} \tilde{b}_i^T & 0 \end{bmatrix}, \quad i = 1, \dots, n,$$

where  $\tilde{b}_i \in \mathbf{R}^{k_i}$  is a vector of the first  $k_i$  (possibly nonzero) entries of  $b_i$ . Since  $B$  is a left inverse of  $A$ , we must have

$$A^T \begin{bmatrix} \tilde{b}_i \\ 0 \end{bmatrix} = e_i, \quad i = 1, \dots, n,$$

which can be written as

$$\tilde{A}^{(i)T} \tilde{b}_i = e_i, \quad i = 1, \dots, n,$$

where  $\tilde{A}^{(i)} \in \mathbf{R}^{k_i \times n}$  is a matrix of the first  $k_i$  rows of  $A$ . Thus  $B$  exists if and only if

$$e_i \in \mathcal{R}(\tilde{A}^{(i)T}), \quad i = 1, \dots, n.$$

Notice that this condition depends on both  $A$  and the deadlines  $k_1, \dots, k_n$ .

An example of a sufficient (but not necessary) condition is,

$$k_1 \geq n, \text{ and } \mathbf{Rank}(\tilde{A}^{(1)}) = n.$$

- (b) We can write

$$J = \sum_{i=1}^n \|\tilde{b}_i\|^2.$$

To minimize  $J$  we can find the least norm solution of each of the equations

$$\tilde{A}^{(i)T} \tilde{b}_i = e_i, \quad i = 1, \dots, n.$$

Assuming  $\tilde{A}^{(1)}, \dots, \tilde{A}^{(n)}$  are full rank, the least norm solution is

$$\tilde{b}_i = \tilde{A}^{(i)} (\tilde{A}^{(i)T} \tilde{A}^{(i)})^{-1} e_i, \quad i = 1, \dots, n.$$

(c) The following matlab code solves this problem.

```
unbdl_data;

% solve for B with deadline constraints
B = zeros(n,m);
for i = 1:n
    ei = zeros(n,1); ei(i) = 1;
    B(i,:) = [pinv(A(1:k(i),:))'*ei; zeros(m-k(i),1)]';
end

% solve for B without deadline constraints
Bnd = pinv(A);

JB = sum(sum(B.^2))
JBnd = sum(sum(Bnd.^2))
```

The value of  $J$  for our estimator (with deadline constraints) is 1.0697 while the value of  $J$  for  $B = A^\dagger$  is 0.1499.