EE263 Midterm Exam

This is a 24 hour take-home midterm. Please turn it in at Bytes Cafe in the Packard building, 24 hours after you pick it up.

- You may use any books, notes, or computer programs (*e.g.*, matlab), but you may not discuss the exam with others until Oct. 26, after everyone has taken the exam. The only exception is that you can ask the TAs or Professor Lall for clarification, by emailing to the staff email address `ee263-aut1415-staff@lists.stanford.edu`. We've tried pretty hard to make the exam unambiguous and clear, so we're unlikely to say much.

- Since you have 24 hours, we expect your solutions to be legible, neat, and clear. Do not hand in your rough notes, and please try to simplify your solutions as much as you can. We will deduct points from solutions that are technically correct, but much more complicated than they need to be.

- Please check your email a few times during the exam, just in case we need to send out a clarification or other announcement. It's unlikely we'll need to do this, but you never know.

- Attach the official exam cover page (available when you pick up or drop off the exam, or from the midterm info page) to your exam, and assemble your solutions to the problems in order, *i.e.*, problem 1, problem 2, ..., problem 6. Start each solution on a new page.

- Please make a copy of your exam before handing it in. We have never lost one, but it might occur.

- When a problem involves some computation (say, using matlab), we do not want just the final answers. We want a clear discussion and justification of exactly what you did, the matlab source code that produces the result, and the final numerical result. Be sure to show us your verification that your computed solution satisfies whatever properties it is supposed to, at least up to numerical precision. For example, if you compute a vector x that is supposed to satisfy $Ax = b$ (say), show us the matlab code that checks this, and the result. (This might be done by the matlab code `norm(A*x-b)`; be sure to show us the result, which should be very small.) *We will not check your numerical solutions for you, in cases where there is more than one solution.*

- In the portion of your solutions where you explain the mathematical approach, you *cannot* refer to matlab operators, such as the backslash operator. (You can, of course, refer to inverses of matrices, or any other standard mathematical construct.)

- Some of the problems are described in a practical setting, such as energy consumption, population dynamics, or wireless communications. *You do not need to understand anything about the application area to solve these problems.* We've taken special care to make sure all the information and math needed to solve the problem is given in the problem description.

- Some of the problems require you to download and run a matlab file to generate the data needed. These files can be found at the URL

  `http://www.stanford.edu/class/ee263/mterm14_mfiles/FILENAME`

  where you should substitute the particular filename (given in the problem) for `FILENAME`. *There are no links on the course web page pointing to these files, so you'll have to type in the whole URL yourself.*

1. Some true/false questions. For each part, specify whether the statement is true or false. Do not give any explanation, proofs, or counterexamples; we will grade any question as incorrect if it contains any explanation.

a) If Q has orthonormal columns then $\|Q^T w\| \leq \|w\|$ for all w.

b) Suppose $A \in \mathbb{R}^{m\times p}$ and $B \in \mathbb{R}^{m\times q}$. If $\mathcal{N}(A) = \{0\}$ and $\mathcal{R}(A) \subseteq \mathcal{R}(B)$ then $p \leq q$.

c) If $V = \begin{bmatrix} V_1 & V_2 \end{bmatrix}$ is invertible and $\mathcal{R}(V_1) = \mathcal{N}(A)$, then $\mathcal{N}(AV_2) = \{0\}$.

d) $\text{rank}([A \quad B]) = \text{rank}(A) = \text{rank}(B) \Longrightarrow \mathcal{R}(A) = \mathcal{R}(B)$.

e) $x \in \mathcal{N}(A^T) \iff x \notin \mathcal{R}(A)$.

f) If A is invertible, then AB is not full rank if and only if B is not full rank.

g) A is not full rank $\Rightarrow$ there is an $x \neq 0$, such that $Ax = 0$.

Solution.

a) **True.**

$$Q = [q_1, \ldots, q_r],$$

suppose $q_i \in \mathbb{R}^n$, we can construct a orthonormal basis of $\mathbb{R}^n$ by adding $n-r$ orthonormal columns to Q. Then

$$\begin{aligned}
\|Q^T w\|^2 &= w^T Q Q^T w \\
&= w^T \left(q_1 q_1^T + \ldots + q_r q_r^T\right) w \\
&= \sum_{i=1}^{r} \left(w^T q_i\right)^2 \\
&\leq \sum_{i=1}^{n} \left(w^T q_i\right)^2 \\
&= w^T \left(q_1 q_1^T + \ldots + q_n q_n^T\right) w \\
&= w^T I w \\
&= w^T w \\
&= \|w\|.
\end{aligned}$$

b) **True.**

$$\mathcal{N}(A) = \{0\} \Leftrightarrow \dim(\mathcal{R}(A)) = p,$$

$$\mathcal{R}(A) \subseteq \mathcal{R}(B) \Leftrightarrow p = \dim(\mathcal{R}(A)) \leq \dim(\mathcal{R}(B)),$$

and since

$$q \geq \text{rank}(B) = \dim(\mathcal{R}(B)),$$

we have

$$q \geq p.$$

c) **True.** $\forall x$ if $x \in \mathcal{N}(AV_2)$, then $V_2 x \in \mathcal{N}(A) = \mathcal{R}(V_1)$, and by definition $V_2 x \in \mathcal{R}(V_2)$ so $V_2 x \in \mathcal{R}(V_1) \cap \mathcal{R}(V_2)$. But V is invertible therefore $\mathcal{R}(V_1) \cap \mathcal{R}(V_2) = 0$, then $V_2 x = 0$. Since V_2 is full rank and skinny therefore $\mathcal{N}(V_2) = \{0\}$, so $x = 0$.

d) **True.**

$\text{rank}([A \quad B]) = \text{rank}(A) \Longrightarrow \mathcal{R}(B) \subseteq \mathcal{R}(A)$.

$\text{rank}([A \quad B]) = \text{rank}(B) \Longrightarrow \mathcal{R}(A) \subseteq \mathcal{R}(B)$.

e) **False.**

$A = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $x = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

f) **True.**

B is not full rank, A invertible $\Longrightarrow AB$ is not full rank obvious.

Define $C = AB$, then $B = A^{-1}C$. Use the first conclusion, $AB = C$ is not full rank, A^{-1} invertible $\Longrightarrow A^{-1}C = B$ is not full rank.

g) **True.** If A fat there is always an x s.t. $Ax = 0$.

If $A \in \mathbb{R}^{m \times n}$ is square or skinny, $m \geq n$. Then if there is an $x \neq 0$, such that $Ax = 0$ then $\dim\mathcal{N}(A) \geq 1$ therefore $\text{rank}(A) \leq n - 1$ which is not full rank.

2. Zeroing out the board. Bobbie and Reza are playing a game. This game is played on a 6×6 board as follows. First, Bobbie fills the board with 36 arbitrary real numbers and then Reza performs a sequence of actions:

At each step, Reza is allowed to choose one cell, and an arbitrary real number x. Then he can add x to the selected cell and subtract x from all adjacent cells. (Some cells have four adjacent cells, some have three, and some have two.) Reza's goal to perform a sequence of allowed actions to derive a table which consists of 36 zeros. If Reza can derive the table consisting of zeros, he wins the game, otherwise Bobbie is the winner.

a) If Bobbie writes 1 in a corner cell (and 0 elsewhere), can Reza win the game? If you believe the answer is positive, you should specify the sequence of actions Reza should take. If your answer is negative, you should prove that there is no possible sequence of actions that Reza can take to zero out the table.

b) Can Bobbie fill in the table so that Reza has no possible way of winning the game? If your answer is positive, you should prove that there exists an initial table that Reza cannot turn into zero. If your answer is negative, you should prove that Reza can turn any initial table into zero with a sequence of allowed actions.

c) Solve part b for a 9×9 table.

Solution.

a) We represent every 6×6 table by a vector in $s \in \mathbb{R}^{36}$. Let $s^{(0)} \in \mathbb{R}^{36}$ represent the initial configuration. The action associated with the real number x and the ith cell is equivalent to adding a vector $xa_i \in \mathbb{R}^{36}$ to s where

$$(a_i)_j = \begin{cases} 1 & j \text{ denotes the selected cell } i \\ -1 & j \text{ denotes an adjacent cell to the selected cell } i \\ 0 & \text{Otherwise.} \end{cases}$$

Thus, the table derived after a sequence of actions will be $s^{(0)} + \sum_i x_i a_i = s^{(0)} + Ax$ where A is a matrix whose columns are a_i's. So we are looking for an input x such that $s^{(0)} + Ax = 0$. The following code solves the problem

```
d=6
A=[];
for i=1:d
    for j=1:d
        a=zeros(d^2,1);
        a(d*(i-1)+j)=1;
        if i>1
            a(d*(i-2)+j)=-1;
        end
        if i<d
            a(d*(i)+j)=-1;
        end
        if j>1
            a(d*(i-1)+j-1)=-1;
        end
        if j<d
            a(d*(i-1)+j+1)=-1;
        end
        A=[A,a];
    end
end

s = zeros(d*d,1);
s(1) = -1;
if rank([A,s]) == rank(A)
```

```
    reshape(A\s,d,d)
end
```

We see that the answer will be

$$M = -\frac{1}{13}\begin{bmatrix} 17 & 2 & -9 & -6 & 1 & 3 \\ 2 & -6 & -5 & 2 & 4 & 2 \\ -9 & -5 & 8 & 9 & -1 & -5 \\ -6 & 2 & 9 & 0 & -9 & -6 \\ 1 & 4 & -1 & -9 & -2 & 8 \\ 3 & 2 & -5 & -6 & 8 & 16 \end{bmatrix}.$$

b) The problem is equivalent to verifying that the linear transformation defined by Ax is onto. Since A is square, it suffices to see if A has zero nullspace. The following code solves the problem.

```
clc
for d=[6,9]
    A=[];
    for i=1:d
        for j=1:d
            a=zeros(d^2,1);
            a(d*(i-1)+j)=1;
            if i>1
                a(d*(i-2)+j)=-1;
            end
            if i<d
                a(d*(i)+j)=-1;
            end
            if j>1
                a(d*(i-1)+j-1)=-1;
            end
            if j<d
                a(d*(i-1)+j+1)=-1;
            end
            A=[A,a];
        end
    end
    fprintf('for d=%d, the dimension of nullspace is %d\n',d,d^2-rank(A))
end
```

We see that Reza can always win, since the transformation is onto.

c) The code from part b shows that the transformation is not onto, so Bobbie can select a good initialization to win.

3. Coin collector robot. Consider a robot with unit mass which can move in a frictionless two dimensional plane. The robot has a constant unit speed in the y direction (towards north), and it is designed such that we can only apply force in the x direction. We will apply a force at time t given by f_j for $2j-2 \leq t < 2j$ where $j = 1, \ldots, n$, so that the applied force is constant over time intervals of length 2. The robot is at the origin at time $t = 0$ with zero velocity in the x direction.

There are $2n$ coins in the plane and the goal is to design a sequence of input forces for the robot to collect the maximum possible number of coins. The robot is designed such that it can collect the ith coin only if it exactly passes through the location of the coin (x_i, y_i). In this problem, we assume that $y_i = i$.

a) Find the coordinates of the robot at time t, where t is a positive integer. Your answer should be a function of t and the vector of input forces $f \in \mathbf{R}^n$.

b) Given a sequence of k coins $(x_1, y_1), \ldots, (x_{2n}, y_{2n})$, describe a method to find whether the robot can collect them.

c) For the data provided in `robot_coin_collector.m`, show that the robot cannot collect all the coins.

d) Suppose that there is an arrangement of the coins such that it is not possible for the robot to collect all the coins. Suggest a way to check if the robot can collect all but one of the coins.

e) Run your method on data in `robot_coin_collector.m` and report which coin cannot be collected. Report the input that results in collecting $2n-1$ coins. Plot the location of the coins and the location of the robot at integer times.

Solution.

a) The second coordinate at time t is simply equal to t.

Consider $A \in \mathbb{R}^{2n \times n}$ such that

$$A_{ij} = \begin{cases} 1 & j = \lfloor \frac{i+1}{2} \rfloor \\ 0 & \text{Otherwise.} \end{cases}$$

Then we will have $Af = [f_1, f_1, f_2, \cdots, f_n]$. Similar to the mass/force example, the first coordinate at time t will be equal to $b_t^T Af$ where

$$b_t = [t - \frac{1}{2}, \cdots, \frac{1}{2}, 0, \cdots, 0]^T.$$

b) According to part a, the only possible time to collect the ith coin is at time $t = y_i = i$. Define l_i to be the first coordinate of the location of the robot at time $t = i$. From part a, we see that

$$l_i = b_t^T Af.$$

Let $B \in \mathbb{R}^{n \times n}$ be a matrix whose ith column is b_i and define $C = B^T A$. Then we will have $l = Cf$.

Hence, we see that the necessary and sufficient condition to collect all the coins is that $x \in \text{range}(C)$. This can be simply examined with `rank([C x])==rank(C)`.

c) The code to solve parts c,e can be find at the bottom.

d) In part b, we saw that $l = Cf$. We know that there exists a sequence of input forces f such that all but one of the $2n$ equations are satisfied, but we don't know which one.

Let $x^{(i)}$ be the location vector x with the ith entry removed. Likewise, let $C^{(i)}$ be the transition matrix with the ith row of C removed. If we can collect all coins but the ith one, then we will certainly have $x^{(i)} \in \text{range}(C^{(i)})$. We will loop over the coins and see whether it's possible to collect all coins but one.

e) The following code solves the problem:

```
clc
clear all
close all
robot_coin_collector

BT = zeros(2*n,2*n);
for i=1:2*n
    BT(i,1:i) = i-1/2:-1:1/2;
end

A = zeros(2*n,n);
for i=1:n
    A([2*i-1,2*i],i)=1;
end
C = BT*A;

%part c
if rank([C,x])==rank(C)
    fprintf('All coins can be collected!\n')
else
    fprintf('All coins cannot be collected!\n')
end

%part e
for i=1:2*n
    xt = x([1:i-1,i+1:end]);
    Ct = C([1:i-1,i+1:end],:);
    if rank([Ct,xt])==rank(Ct)
        fprintf('The robot can collect all coins but %dth,\n',i);
        fprintf('and the input will be: \n')
        input = Ct\xt;
        disp(input)
    end
end
```

```
hold on
plot(C*input,1:2*n,'r')
hold off
```

We see that all coins but the 7^{th} can be collected and the associated input will be

$$f = [1.0000, -4.0000, 7.0000, -10.0000, 20.0000, -35.0000].$$

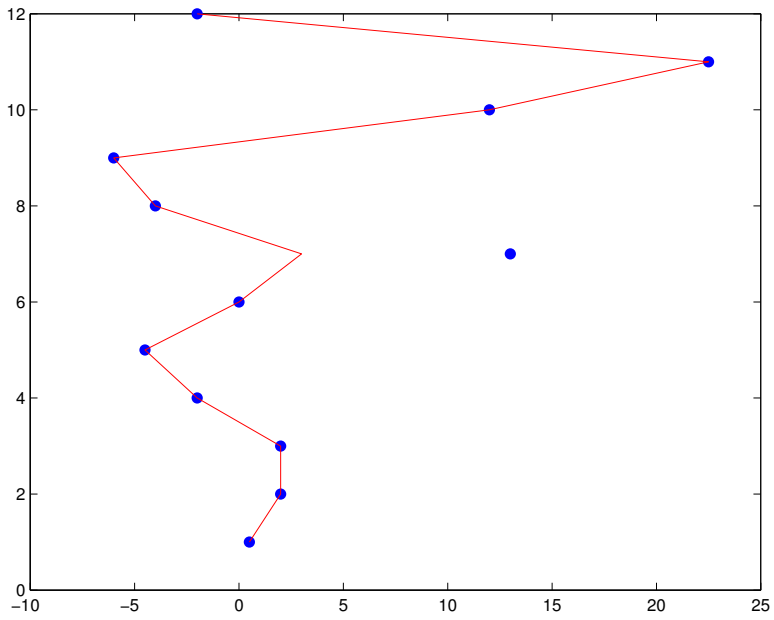


Figure 1: Location of the coins and the trajectory of the robot

4. Power generation for a city. A city has a series of generators that provide power to several key sites. Each site submits an estimate of the amount of power it plans to use for the year. Certain generators are more efficient for providing power to some of the sites than others. The following table provides a list of the generators and the amount of power it can generate per unit cost for each site.

	Generator 1	Generator 2	Generator 3	Generator 4
Site A	5	2	1	0
Site B	1	1	2	3
Site C	0	1	2	4
Site D	0	3	3	1
Site E	3	2	1	1
Site F	2	0	3	1
Site G	1	3	0	2

Here is the amount of power each site estimates it will use for the year:

Site A	Site B	Site C	Site D	Site E	Site F	Site G
20	5	3	4	10	5	5

The goal is to estimate how much money will be spent at each generator while trying to meet the sites' estimates. There is one requirement, however. Site A, a hospital, has been given top priority, and therefore must receive the **exact** amount of power it has requested. The other sites are allowed to receive a little more or less than the power it requested, though the goal is still to get as close as possible to meeting every site's request.

To avoid typos, the generator table and site requirements can be found at `powergen.m`.

Here is what we want you to answer. Be sure to explain your process and how you arrived at your answer.

a) Find the vector of money spent for each generator that minimizes the sum of the squared deviation between the sites' power requested and power generated, provided that Site A receives its exact estimate.

b) Generator 4 has stopped working. Find the tradeoff curve showing how close the other sites can get to their estimates as the requirement for Site A is relaxed. Plot the sum of the squared deviation of what Sites B through G request and receive versus the difference squared between what Site A requests and what it receives.

Solution.

a) The set of solutions which can satisfy the constraint is the set of x for which the following is true:

$$20 = [5\ 2\ 1\ 0]x.$$

We can write this set as

$$x = \{x_0 + z : z \in \mathcal{N}([5\ 2\ 1\ 0])\},$$

where x_0 is an arbitrary solution, (i.e. $x_0 = [4\ 0\ 0\ 0]^T$). Let N be a basis for $\mathcal{N}([5\ 2\ 1\ 0])$. Then $x = x_0 + Nw$, where w is the vector we have control over. Thus, we have the following least squares problem.

$$\text{Minimize} \qquad ||\widetilde{A}(x_0 + Nw) - \widetilde{y}||^2,$$

where $\widetilde{A}$ is the matrix A with the top row removed and $\widetilde{y}$ is the vector y with the first entry removed. Rearranging some terms around, we have

$$\text{Minimize} \qquad ||\widetilde{A}Nw - (\widetilde{y} - \widetilde{A}x_0)||^2,$$

and so

$$w = ((\widetilde{A}N)^T(\widetilde{A}N))^{-1}(\widetilde{A}N)^T(\widetilde{y} - \widetilde{A}x_0),$$

and

$$x = x_0 + Nw.$$

This gives us the following cost vector:

$$x = \begin{bmatrix} 3.6084 \\ 0.9771 \\ 0.0039 \\ 0.0702 \end{bmatrix}$$

b) This is a straightforward least squares problem with a weighted sum objective. We first remove the last column of the A matrix since Generator 4 no longer works. Define

$$\hat{A} = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 3 & 3 \\ 3 & 2 & 1 \\ 2 & 0 & 3 \\ 1 & 3 & 0 \end{bmatrix}, \qquad \hat{a} = [5\ 2\ 1], \qquad \hat{y} = \begin{bmatrix} 5 \\ 3 \\ 4 \\ 10 \\ 5 \\ 5 \end{bmatrix}.$$

Our least squares objective function becomes

$$||\hat{a}x - 20||^2 + \mu||\hat{A}x - \hat{y}||^2 = \left|\left|\begin{bmatrix} \hat{a} \\ \sqrt{\mu}\hat{A} \end{bmatrix} x - \begin{bmatrix} 20 \\ \sqrt{\mu}\hat{y} \end{bmatrix}\right|\right|^2$$

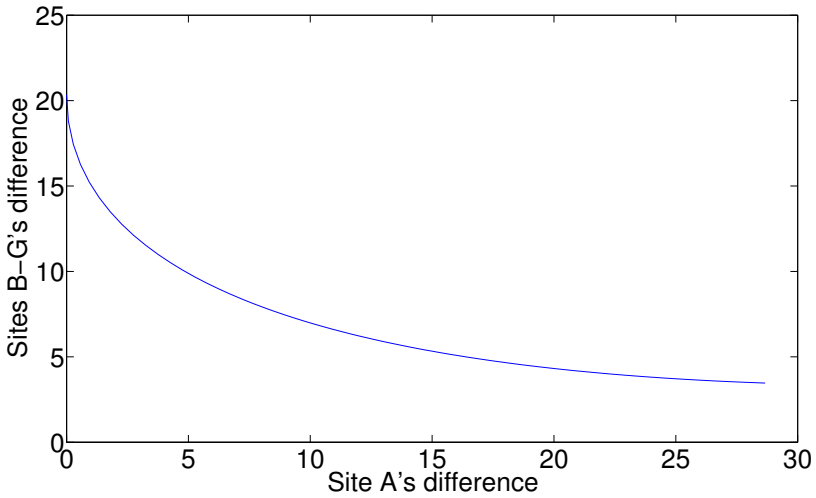
.

We then loop over values of μ to generate the tradeoff curve.

The following is Matlab code which calculates the solution for parts (a) and (b).

```
powerGen

% Part a
x0 = [4;0;0;0];
N  = null([5 2 1 0]);

A_tilde = A(2:end, :);
y_tilde = y(2:end);

w = (A_tilde*N) \ (y_tilde - A_tilde*x0);
x_constrained = x0 + N*w

% Part b
A_hat = A(2:end,1:3);
a_hat = A(1,1:3);
y_hat = y(2:end);

mu = [0:0.1:20];
for ind = 1:length(mu)
  % Solve constrained least squares when mu = 0
```

```
  if (ind == 1)
    x0 = [4;0;0];
    N  = null([5 2 1]);
    A_tilde = A(2:end, 1:3);
    y_tilde = y(2:end);

    w = (A_tilde*N) \ (y_tilde - A_tilde*x0);
    x = x0 + N*w;
  % Multi-object least squares when mu > 0
  else
    newA = [a_hat; sqrt(mu(ind))*A_hat];
    newy = [20; sqrt(mu(ind))*y_hat];
    x = (newA)\newy;
  end

  obj1(ind) = (a_hat*x - 20)^2;
  obj2(ind) = sum((A_hat*x - y_hat).^2);
end

plot(obj1, obj2);
xlabel('Site A''s difference');
ylabel('Sites B-G''s difference');

% Output:
% x_constrained =
%
%     3.6084
%     0.9771
%     0.0039
%     0.0702
```



5. Estimating a communication channel. In a communications channel, the input $x_1, \ldots, x_N$ and output $y_1, \ldots, y_N$ are related by the following equation:

$$y_n = \sum_{m=1}^{M} h_m x_{n-m+1} + v_n \qquad \text{for } n = 1, \ldots, N,$$

where $h_1, \ldots, h_M$ is the impulse response of the channel and $v_1, \ldots, v_N$ is some noise term. In order to estimate the impulse response of the channel, it is typical to send a known input sequence through the channel, and from the output deduce the channel coefficients.

Our task is to estimate the channel which minimizes the following objective function:

$$J = \sum_{n=1}^{N} \left(y_n - \sum_{m=1}^{M} h_m x_{n-m+1} \right)^2.$$

Assume that $x_n = 0$ for $n \leq 0$.

One thing that we have left out is the length of the impulse response. We can get around this problem by iterating over the lengths of the channel and observing how the residual behaves.

There are three parts to this question:

a) Using the data provided in `channel.m`, find the impulse response of the channel assuming the length of the impulse response is 4. The known input and output signals are stored in the variables `x_known` and `y_known`, respectively. Explain how you arrived at those coefficients.

b) Iterate your solution from part (a) assuming the length of the impulse response is $3, \ldots, 10$ and give the corresponding residuals. Plot the residual J as a function of the impulse response length. Is 4 a reasonable estimate? What would you recommend as a good length for the impulse response?

c) You receive more output from the channel (stored in the variable `y_unknown`), except now you don't know what the input signal is. y_n runs from $n = 1, \ldots, N + 10$, although $x_n = 0$ for $n > N$. Using the channel you found in part (b), find and make a stem plot (using Matlab's `stem` command) of your estimated signal x_n for $n = 1, \ldots, N$. Also report the residual.

Solution.

a) Using the equation for y_n given to us, we have

$$\begin{aligned}
y_1 &= h_1 x_1 \\
y_2 &= h_1 x_2 + h_2 x_1 \\
y_3 &= h_1 x_3 + h_2 x_2 + h_3 x_1 \\
y_4 &= h_1 x_4 + h_2 x_3 + h_3 x_2 + h_4 x_1 \\
y_5 &= h_1 x_5 + h_2 x_4 + h_3 x_3 + h_4 x_2 \\
&\vdots \\
y_{40} &= h_1 x_{40} + h_2 x_{39} + h_3 x_{38} + h_4 x_{37}
\end{aligned}$$

Thus, we can express our input/output relation as

$$y = \underbrace{\begin{bmatrix} x_1 & 0 & 0 & 0 \\ x_2 & x_1 & 0 & 0 \\ x_3 & x_2 & x_1 & 0 \\ x_4 & x_3 & x_2 & x_1 \\ x_5 & x_4 & x_3 & x_2 \\ x_6 & x_5 & x_4 & x_3 \\ x_7 & x_6 & x_5 & x_4 \\ & & \vdots & \\ x_{40} & x_{39} & x_{38} & x_{37} \end{bmatrix}}_{A} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix}.$$

The channel is given by $h = (A^T A)^{-1} A^T y$, which gives us

$$h = \begin{bmatrix} 0.1721 \\ -0.0958 \\ 0.0080 \\ 1.1205 \end{bmatrix}$$

b) As we increase the length of the channel, we simply add another column to our matrix A. We see in the figure that while a length 4 channel is a big improvement over a length 3 channel, a length 7 channel seems to be a good length. (Increasing the channel beyond that gives minimal improvement). The length 7 filter is:

$$h = \begin{bmatrix} 0.3141 \\ -0.1672 \\ -0.0114 \\ 1.1192 \\ -0.0166 \\ -0.2711 \\ 0.2955 \end{bmatrix}$$

c) The roles of h and x are reversed. Our matrix will now contain the values of the filter we recovered from part (b), and will be skinny (since the output y is longer than the input x). Here is what the setup looks like

$$y = \begin{bmatrix} h_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ h_2 & h_1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ h_3 & h_2 & h_1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ h_4 & h_3 & h_2 & h_1 & 0 & 0 & 0 & 0 & \dots & 0 \\ h_5 & h_4 & h_3 & h_2 & h_1 & 0 & 0 & 0 & \dots & 0 \\ h_6 & h_5 & h_4 & h_3 & h_2 & h_1 & 0 & 0 & \dots & 0 \\ h_7 & h_6 & h_5 & h_4 & h_3 & h_2 & h_1 & 0 & \dots & 0 \\ 0 & h_7 & h_6 & h_5 & h_4 & h_3 & h_2 & h_1 & \dots & 0 \\ & & \vdots & & & & & & & \\ 0 & & & & & & & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_{39} \end{bmatrix}.$$

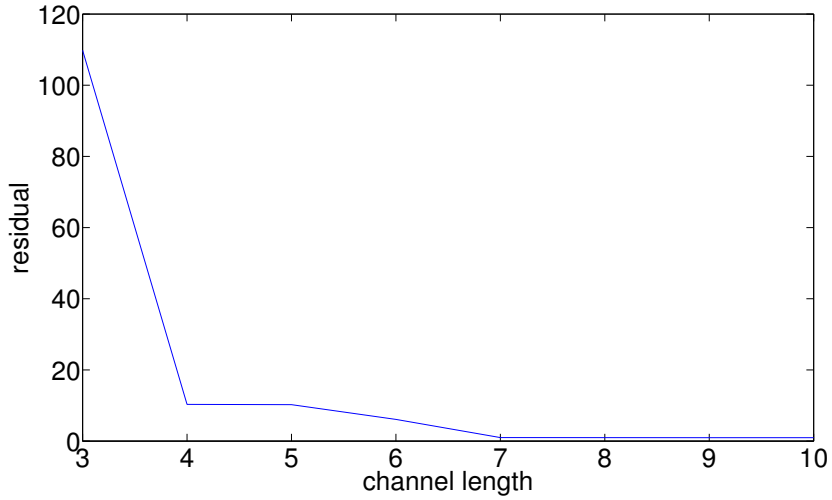
Note that the last few rows of the matrix are zero because the filter length is only 7. The residual we get is 0.4962.

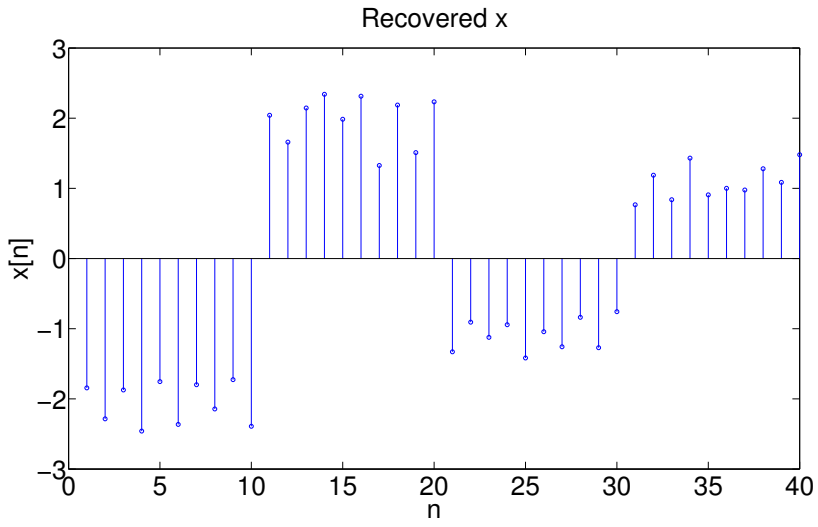
The following is Matlab code which calculates the solution.

```
channel;

% Part a
M = 4;
A = zeros(N,M);
for ind = 1:M
  A(:,ind) = [zeros(ind-1,1);x_known(1:end-(ind-1))];
end
h4 = A\y_known

% Part b
residuals = zeros(10,1);
for M = 3:10;
  A = zeros(N,M);
  for ind = 1:M
    A(:,ind) = [zeros(ind-1,1);x_known(1:end-(ind-1))];
  end

  hGuess = A\y_known;

  residuals(ind)= sum((y_known - A*hGuess).^2);
end
residuals(3:10)

% Part c
% 7 seems to be reasonable, so we recover that filter
M = 7;
A = zeros(N,M);
for ind = 1:M
  A(:,ind) = [zeros(ind-1,1);x_known(1:end-(ind-1))];
end
hBest = A\y_known

% Now recover x from y_unknown
H = zeros(N+10,N);
column = zeros(N+10,1);
column(1:M) = hBest;
for ind = 1:N
  H(:,ind) = [zeros(ind-1,1);column(1:end-(ind-1))];
end
x_recovered = H\y_unknown;
```

```
residual = sum((y_unknown - H*x_recovered).^2)

fSize = 40;
figure(1)
plot([3:10], residuals(3:10));
xlabel('channel length', 'fontSize', fSize);
ylabel('residual', 'fontSize', fSize);
set(gca,'fontSize', fSize)

figure(2)
stem(x_recovered);
title('Recovered x', 'fontSize', fSize);
xlabel('n', 'fontSize', fSize);
ylabel('x[n]', 'fontSize', fSize);
set(gca,'fontSize', fSize)

% Output
% h4 =
%
%     0.1721
%    -0.0958
%     0.0080
%     1.1205
%
%
% ans =
%
%   109.8208
%    10.3245
%    10.2338
%     6.0975
%     0.9878
%     0.9672
%     0.9635
%     0.9592
%
%
% hBest =
%
%     0.3141
%    -0.1672
%    -0.0114
%     1.1192
%    -0.0166
%    -0.2711
```

```
%     0.2955
%
%
% residual =
%
%     0.4962
```





6. Heating lamp power control. Heating process is a usually critical process in chemical synthesis. A heating chamber usually consists of several lamps for heating and several temperature sensors (for example, thermocouples) for monitoring. It is often required that the temperature increase rate is constant, but unfortunately, in reality, the temperature increase rate is a nonlinear function with lamp powers, current temperature distribution, and the position of lamps, etc.

To achieve active lamp power control, we could

- solve heating transfer differential equations to get the nonlinear relationship between the temperature increase rate and lamp powers and current temperature profiles. But due to the environment's fluctuations and the time consuming of solving PDE problems, this method usually does not work well.
- Assume the relationship between temperature increase rate and lamp powers and current temperature is linear in a small time and temperature interval. Then we can solve the dynamics of the smaller scale linear system very fast and use the dynamics to generate next power outputs, based on target temperature increase profile.

Now we consider a even simpler model: the temperature increase rate of each sensors is a linear function of the lamp powers all the time.

Here is the problem. In a heating system in 2D space, there are 3 lamps located in different positions and there are 4 thermocouples to monitor the temperature in the heating area. We already took the temperature measurement and recorded lamp power outputs.

You are given these records

$$p(1), \ldots, p(N), \quad T(1), \ldots, T(N),$$

where $p(t) \in \mathbb{R}^3$, are the power outputs of the 3 lamps, and $T \in \mathbb{R}^4$ are the measured temperatures from the 4 thermocouples.

Now we assume at each discrete time t,

$$T(t) - T(t-1) = A(t)p(t).$$

Where the matrix $A(t) \in \mathbb{R}^{4\times 3}$ is the parameter matrix of the model. Since now we are considering the simplest model, we assume all $A(t)$ are the same, so we have

$$T(t) - T(t-1) \approx Ap(t).$$

The method works by choosing A that minimizes the RMS (root-mean-square) of the prediction error over $t = 2, 3, \ldots, N$. In another words, we want to choose the matrix A to minimize

$$\Big(\frac{1}{N-1}\sum_{t=2}^{N} \|T(t) - T(t-1) - Ap(t)\|_2^2\Big)^{1/2}$$

a) Explain how to obtain A, given the records $T(t)$ and $p(t)$.

b) Implement the method, and apply it to the data given in the file `temp_control_data.m`. This file contains a temperature record matrix

$$T = \begin{bmatrix} T(1) & T(2) & \ldots & T(N) \end{bmatrix}$$

and a power supply record matrix

$$p = \begin{bmatrix} p(1) & p(2) & \ldots & p(N) \end{bmatrix}$$

Find the matrix A which minimizes the RMS of the prediction error, and compute the RMS prediction error.

Solution. The prediction error at time t is

$$e(t) = T(t) - T(t-1) - Ap(t),$$

We can express matrix A in terms of rows

$$A_j = \begin{bmatrix} \tilde{a}_1^T \\ \vdots \\ \tilde{a}_4^T \end{bmatrix},$$

Let $e_i(t)$ be the ith component of the vector $e(t)$. It is given by

$$e_i(t) = T_i(t) - T_i(t-1) - \tilde{a}_i^T p(t), \quad i = 1, 2, 3, 4.$$

The RMS value of the prediction could be written as

$$\begin{aligned} \Big(\frac{1}{N-1}\sum_{t=2}^{N} \|T(t) - T(t-1) - Ap(t)\|_2^2\Big)^{1/2} &= \Big(\frac{1}{N-1}\sum_{t=2}^{N}\sum_{i=1}^{4} e_i(t)^2\Big)^{1/2} \\ = \Big(\frac{1}{N-1}\sum_{i=1}^{4}\sum_{t=2}^{N} e_i(t)^2\Big)^{1/2} &= \Big(\frac{1}{N-1}\sum_{i=1}^{4} \|e_i\|_2^2\Big)^{1/2} \end{aligned}$$

Where e_i denotes the i th prediction error component for all time instance. The first operation follows from the definition of the l_2 norm of the error vector at a particular time instance. After changing the order of summation you obtain the expression for the l_2 norm of the time course of a single component e_i. This shows that the problem is in fact separable, the minimization of the global error is equivalent to independent minimization of the errors due to each component.

The l_2 norm of the ith error component is

$$||e_i||^2 = ||P\tilde{a}_i - \delta T_i||^2.$$

where

$$P = \begin{bmatrix} p^T(2) \\ p^T(3) \\ \vdots \\ p^T(N) \end{bmatrix}, \delta T_i = \begin{bmatrix} T_i(2) - T_i(1) \\ T_i(3) - T_i(2) \\ \vdots \\ T_i(N) - T_i(N-1) \end{bmatrix}$$

for $i = 1, 2, 3, 4$. The ith error component is minimized by the least-squares approximate solution, which, if matrix P is full rank, is given by

$$\tilde{a}_{ils} = (P^T P)^{-1} P^T \delta T_i, \quad i = 1, 2, 3, 4.$$

By defining $\delta T = [\delta T_1, \ldots, \delta T_4]$, we have

$$A_{ls}^T = [\tilde{a}_{1ls}, \ldots, \tilde{a}_{4ls}] = (P^T P)^{-1} P^T \delta T,$$

so

$$A_{ls} = ((P^T P)^{-1} P^T \delta T)^T.$$

Execution of the code below produces the following A estimation.

$$A_{ls} = \begin{bmatrix} 0.0235 & 0.0152 & 0.0088 \\ 0.0214 & 0.0168 & 0.0094 \\ 0.0221 & 0.0165 & 0.0089 \\ 0.0195 & 0.0171 & 0.0110 \end{bmatrix}$$

associated RMS value of the prediction error is 0.1084.

```
clear all;
delta_T=(T(:,2:N)-T(:,1:N-1))';
P=p(:,2:N)';
%Calculate least squares approximate solution
A_ls=(inv(P'*P)*P'*delta_T)';
%Calculate RMS error value
RMS=sqrt(1/(N-1)*sum(sum((T(:,2:N)-T(:,1:N-1)-A_ls*p(:,2:N)).^2)));
```