1. Memory of a linear, time-invariant system. Suppose an input signal $(u_t : t \in \mathbb{Z})$, and an output signal $(y_t : t \in \mathbb{Z})$ are related by a convolution operator:

$$y_t = \sum_{\tau=1}^{M} h_{\tau} u_{t-\tau},$$

where $h = (h_1, \ldots, h_M)$ are the impulse-response coefficients of the convolution system. (Convolution systems are also called linear, time-invariant systems.) If $h_M \neq 0$, then $M$ is called the memory of the system. You are given the input and output signals for $t = 1, \ldots, T$:

$$u_1, \ldots, u_T \quad \text{and} \quad y_1, \ldots, y_T.$$

However, you do not know $u_t$ or $y_t$ for $t < 1$ or $t > T$, and you do not know the impulse response, $h$.

a) Explain how to find the smallest value of $M$, and a corresponding impulse response $(h_t : t = 1, \ldots, M)$ that is consistent with the given data. You may assume that $T > 2M$.

b) Apply your method to the data in lti_memory_data.m. Report the value of $M$ that you find.

Hint. The function `toeplitz` may be useful.

Solution.

a) We have data for the observations

$$y_t = \sum_{\tau=1}^{M} h_{\tau} u_{t-\tau}, \quad t = M + 1, \ldots, T.$$

We can collect the data into a single matrix-vector equation:

$$\begin{bmatrix} y_{M+1} \\ y_{M+2} \\ \vdots \\ y_T \end{bmatrix} = \begin{bmatrix} u_{M} & u_{M-1} & \cdots & u_1 \\ u_{M+1} & u_{M} & \cdots & u_2 \\ \vdots & \vdots & \ddots & \vdots \\ u_{T-1} & u_{T-2} & \cdots & u_{T-M} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_M \end{bmatrix}. $$

1
A model that has memory $M$ is consistent with the observed data $(u_t : t = 1, \ldots, T)$ and $(y_t : t = 1, \ldots, T)$ if the system given above has a solution $(h_t : t = 1, \ldots, M)$. Then, $(h_t : t = 1, \ldots, M)$ is an impulse response that is consistent with the data. In order to find the smallest value of $M$, we simply try $M = 1, 2, \ldots$, and take the smallest value of $M$ that is consistent with the data.

b) The smallest memory consistent with the data is $M = 7$. The script used to perform the analysis is given below.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% clean up the workspace, and load the data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all; clc
lti_memory_data;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% find the smallest value of M consistent with the data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for M = 1:(T-1)
    yM = y((M+1):T);
    UM = toeplitz(u(M:(T-1)) , u(M:(-1):1));
    if rank(UM) == rank([UM yM])
        break;
    end
end
hM = UM \ yM;
fprintf('M = %d
' , M);
```

2. Norm preserving implies orthonormal columns. In lecture we saw that if $A \in \mathbb{R}^{m \times n}$ has orthonormal columns, i.e., $A^T A = I$, then for any vector $x \in \mathbb{R}^n$ we have $\| A x \| = \| x \|$. In other words, multiplication by such a matrix preserves norm.

Show that the converse holds: If $A \in \mathbb{R}^{m \times n}$ satisfies $\| A x \| = \| x \|$ for all $x \in \mathbb{R}^n$, then $A$ has orthonormal columns (and in particular, $m \geq n$).

*Hint.* Start with $\| A x \|^2 = \| x \|^2$, and try $x = e_i$, and also $x = e_i + e_j$, for all $i \neq j$.

**Solution.** Suppose that $\| A x \| = \| x \|$ for all $x$. Then

$$\| A x \|^2 = x^T (A^T A) x = \| x \|^2 = x^T x$$

for all $x$. Let’s start with $x = e_i$. The equation above then reduces to

$$e_i^T (A^T A) e_i = (A^T A)_{ii} = 1,$$
and we see that the diagonal entries in $A^T A$ are all one. Now let's plug in $x = e_i + e_j$, to get

$$(e_i + e_j)^T (A^T A)(e_i + e_j) = e_i^T (A^T A)e_i + e_j^T (A^T A)e_j + e_i^T (A^T A)e_j + e_j^T (A^T A)e_i$$

$$= 2 + 2e_j^T (A^T A)e_i$$

$$= (e_i + e_j)^T(e_i + e_j)$$

$$= 2.$$

We conclude that $e_j^T (A^T A)e_i = (A^T A)_{ij} = 0$, for $j \neq i$. Thus, the off-diagonal elements of $A^T A$ are zero. So we have $A^T A = I$.

### 3. Sensor integrity monitor

A suite of $m$ sensors yields measurement $y \in \mathbb{R}^m$ of some vector of parameters $x \in \mathbb{R}^n$. When the system is operating normally (which we hope is almost always the case) we have $y = Ax$, where $m > n$. If the system or sensors fail, or become faulty, then we no longer have the relation $y = Ax$. We can exploit the redundancy in our measurements to help us identify whether such a fault has occurred. We'll call a measurement $y$ consistent if it has the form $Ax$ for some $x \in \mathbb{R}^n$. If the system is operating normally then our measurement will, of course, be consistent. If the system becomes faulty, we hope that the resulting measurement $y$ will become inconsistent, i.e., not consistent. (If we are really unlucky, the system will fail in such a way that $y$ is still consistent. Then we’re out of luck.) A matrix $B \in \mathbb{R}^{k \times m}$ is called an integrity monitor if the following holds:

- $By = 0$ for any $y$ which is consistent.
- $By \neq 0$ for any $y$ which is inconsistent.

If we find such a matrix $B$, we can quickly check whether $y$ is consistent; we can send an alarm if $By \neq 0$. Note that the first requirement says that every consistent $y$ does not trip the alarm; the second requirement states that every inconsistent $y$ does trip the alarm. Finally, the problem. Find an integrity monitor $B$ for the matrix

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 1 & -1 & -2 \\ -2 & 1 & 3 \\ 1 & -1 & -2 \\ 1 & 1 & 0 \end{bmatrix}.$$

Your $B$ should have the smallest $k$ (i.e., number of rows) as possible. As usual, you have to explain what you’re doing, as well as giving us your explicit matrix $B$. You must also verify that the matrix you choose satisfies the requirements. **Hints:**

- You might find one or more of the matlab commands `orth`, `null`, or `qr` useful. Then again, you might not; there are many ways to find such a $B$.
- When checking that your $B$ works, don’t expect to have $By$ exactly zero for a consistent $y$; because of roundoff errors in computer arithmetic, it will be really, really small. That’s OK.
- Be very careful typing in the matrix $A$. It’s not just a random matrix.
Solution. The key challenge in this problem is to restate everything in common linear algebra and matrix terms. We need to find \( B \in \mathbb{R}^{k \times m} \) such that the following hold:

- \( By = 0 \) for any consistent \( y \)
- \( By \neq 0 \) for any inconsistent \( y \)

Let’s analyze the conditions, starting with the first one. The set of consistent measurements is exactly equal to the range of the matrix \( A \); so say that \( By = 0 \) for every consistent \( y \) is the same as saying \( \text{range}(A) \subseteq \text{null}(B) \), i.e., every element in the range of \( A \) is also in the nullspace of \( B \). In terms of matrices, the first condition means that for every \( x \), we have \( BAx = 0 \). That’s true if and only if \( BA = 0 \). (Recall these are matrices, so we can have \( BA = 0 \) without \( A = 0 \) or \( B = 0 \).) We now consider the second condition. To say that every inconsistent \( y \) has \( By \neq 0 \) is equivalent to saying that whenever \( By = 0 \), we have \( y \) is consistent. This is the same as saying \( \text{null}(B) \subseteq \text{range}(A) \). Putting this together with the first condition, we get a really simple condition: \( \text{null}(B) = \text{range}(A) \). In other words, we need to find a matrix \( B \) whose nullspace is exactly equal to the range of \( A \). Now to find such a \( B \) with smallest possible number of rows, we need \( B \) to be full rank. Its rank must be \( m \) minus the dimension of the range of \( A \), i.e., \( m - \text{rank}(A) \). Now that we know what we’re looking for, there are several ways to find such a \( B \), given \( A \). Note that whatever method we end up using we can check that we’ve got a solution by checking that \( BA = 0 \) and \( B \) is full rank. One method relies on the fact from lectures that for any matrix \( C \), \( \text{null}(C) \) and \( \text{range}(C^T) \) are orthogonal complements. It follows that \( \text{null}(B) \) and \( \text{range}(B^T) \) are orthogonal complements, and so are \( \text{range}(A) \) and \( \text{null}(A^T) \). We require that \( \text{null}(B) = \text{range}(A) \), so this means their orthogonal complements are equal, i.e., \( \text{range}(B^T) = \text{null}(A^T) \). In matlab, we can compute a basis for the nullspace of \( A^T \) using the command \( \text{null} \). (In fact \( \text{null} \) gives us an orthonormal basis for the nullspace, but for this problem all we care about is that we get a basis for the nullspace.)

This approach can be implemented with the simple matlab code:

```matlab
A = [ 1 2 1 ; 1 -1 -2; -2 1 3 ; 1 -1 -2; 1 1 0 ]; B = null(A');
B*A
rank(B)
```

The matrix \( BA \) does turn out to be zero for all practical purposes; the entries are very, very small, but nonzero because of roundoff error in computer arithmetic. One subtlety you may or may not have noticed is that \( A \) is not full rank; it has rank 2. In fact, its third column is equal to its second column minus its first column. That’s why we end with \( k = 3 \), and not 2, as you might have expected. Another way to find such a \( B \) uses the full QR factorization of \( A \). If we have QR factorization

\[
A = [Q_1 \ Q_2] \begin{bmatrix} R_1 & 0 \\ 0 & 0 \end{bmatrix},
\]

where \([Q_1 \ Q_2]\) is orthogonal and \( R_1 \) is upper triangular and invertible, then the columns of \( Q_1 \) are an orthonormal basis for the range of \( A \), and the columns of \( Q_2 \) are an orthonormal basis for the orthogonal complement. Therefore we can take \( B = Q_2^T \). This approach can be carried out in matlab via

```matlab
[Q,R]=qr(A);
Q2 = Q(:,[3,4,5]); % get the last three columns of Q
```

...
Two common errors involved the size of $B$. In each case, $B$ satisfies $BA = 0$, so whenever $y$ is consistent, we have $By = 0$. The first error was to have a $B$ that is too small, i.e., has fewer than 3 rows. Such a $B$ doesn’t satisfy the second condition; there are inconsistent $y$’s with $By = 0$. Therefore $B$’s with fewer than 3 rows aren’t integrity monitors. The opposite error, of having $B$ with more than 3 rows, isn’t quite so bad. In this case, your $B$ doesn’t have the minimal number of rows, but it is a real integrity monitor.

4. 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 $i$th 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 $f \in \mathbb{R}^n$.

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

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

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 \(i\)th 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

\[ l_i = b_i^T Af. \]

Let \(B \in \mathbb{R}^{n \times n}\) be a matrix whose \(i\)th 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 \(\text{rank}([C \; x]) = \text{rank}(C)\).

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 \(i\)th entry removed. Likewise, let \(C^{(i)}\) be the transition matrix with the \(i\)th row of \(C\) removed. If we can collect all coins but the \(i\)th 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:

```matlab
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]);
```

6
Ct = C([1:i-1,i+1:end],:);
if rank([Ct,xt]) == rank(Ct)
    fprintf('The robot can collect all coins but %dth,
',i);
    fprintf('and the input will be: 
')
    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 7th 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

5. Solving linear equations via QR factorization. Consider the problem of solving the linear equations \( Ax = y \), with \( A \in \mathbb{R}^{n \times n} \) nonsingular, and \( y \) given. We can use the Gram-Schmidt procedure to compute the QR factorization of \( A \), and then express \( x \) as \( x = A^{-1}y = R^{-1}(Q^Ty) = R^{-1}z \), where \( z = Q^Ty \). In this exercise, you'll develop a method for computing
\[ x = R^{-1}z, \text{ i.e., solving } Rx = z, \text{ when } R \text{ is upper triangular and nonsingular (which means its diagonal entries are all nonzero).} \]

The trick is to first find \( x_n \); then find \( x_{n-1} \) (remembering that now you know \( x_n \)); then find \( x_{n-2} \) (remembering that now you know \( x_n \) and \( x_{n-1} \)); and so on. The algorithm you will discover is called \textit{back substitution}, because you are substituting known or computed values of \( x_i \) into the equations to compute the next \( x_i \) (in reverse order). Be sure to explain why the algorithm you describe cannot fail.

\textbf{Solution.} Suppose that

\[
\begin{bmatrix}
z_1 \\
z_2 \\
\vdots \\
z_n
\end{bmatrix}, \quad
\begin{bmatrix}
r_{11} & r_{12} & \cdots & r_{1n} \\
0 & r_{22} & \cdots & r_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & r_{nn}
\end{bmatrix}, \quad
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n
\end{bmatrix}
\]

Consider the linear equation corresponding to the last \((n)\)th row of \( R \), \textit{i.e.,}

\[ z_n = r_{nn}x_n. \]

Since \( r_{nn} \neq 0 \) we can simply solve for \( x_n \) to get \( x_n = z_n/r_{nn} \). Now consider the linear equation corresponding to the \((n-1)\)th row of \( R \), \textit{i.e.,}

\[ z_{n-1} = r_{(n-1)(n-1)}x_{n-1} + r_{(n-1)n}x_n \]

and since \( r_{(n-1)(n-1)} \neq 0 \) we get

\[ x_{n-1} = \frac{1}{r_{(n-1)(n-1)}} \left( z_{n-1} - r_{(n-1)n}x_n \right) \]

with \( x_n \) found from the previous step. In general, if \( x_n, x_{n-1}, \ldots, x_{i+1} \) are known, \( x_i \) can be derived from the linear equation corresponding to the \( i \)th row of \( R \) as \( (\text{assuming } r_{ii} \neq 0) \)

\[ x_i = \frac{1}{r_{ii}} \left( z_i - \sum_{j=i+1}^{n} r_{ij}x_j \right), \]

where again we rely on \( r_{ii} \neq 0 \), which comes from our assumption that \( A \) is nonsingular. Therefore, the \( x_i \)'s can be computed recursively for \( i = n, n-1, \ldots, 1 \) by \textit{back substitution}. This suggests the following simple algorithm:

\begin{verbatim}
  i := n;
  while i ≥ 1
    if r_{ii} ≠ 0
      x_i := \frac{1}{r_{ii}} \left( z_i - \sum_{j=i+1}^{n} r_{ij}x_j \right);
    else
      unique solution does not exist; break;
  end
  i := i - 1;
end
\end{verbatim}
6. Quadratic extrapolation of a time series, using least-squares fit. We are given a series \( z \) up to time \( t \). We extrapolate, or predict, \( z(t + 1) \) based on a least-squares fit of a quadratic function to the previous ten elements of the series, \( z(t), z(t - 1), \ldots, z(t - 9) \). We'll denote the predicted value of \( z(t + 1) \) by \( \hat{z}(t + 1) \). More precisely, to find \( \hat{z}(t + 1) \), we find the quadratic function \( f(\tau) = a_2 \tau^2 + a_1 \tau + a_0 \) for which

\[
\sum_{\tau=t-9}^{t} (z(\tau) - f(\tau))^2
\]

is minimized. The extrapolated value is then given by \( \hat{z}(t + 1) = f(t + 1) \).

a) Show that

\[
\hat{z}(t + 1) = c \begin{bmatrix} z(t) \\ z(t - 1) \\ \vdots \\ z(t - 9) \end{bmatrix},
\]

where \( c \in \mathbb{R}^{1 \times 10} \) does not depend on \( t \). Find \( c \) explicitly.

b) Use the following matlab code to generate a time series \( z \):

\[
t = 1:1000;
z = 5*\sin(t/10 + 2) + 0.1*\sin(t) + 0.1*\sin(2*t - 5);
\]

Use the quadratic extrapolation method from part (a) to find \( \hat{z}_{ls}(t) \) for \( t = 11, \ldots, 1000 \). Find the relative root-mean-square (RMS) error, which is given by

\[
\left( \frac{(1/990) \sum_{j=11}^{1000} (\hat{z}(j) - z(j))^2}{(1/990) \sum_{j=11}^{1000} z(j)^2} \right)^{1/2}.
\]

c) Plot \( z \) (the true values), and \( \hat{z}_{ls} \) (the estimated values using least-squares), on the same plot. Restrict your plot to \( t = 1, \ldots, 100 \).

Solution.

a) Let \( f \) be the function that used to predict \( z(t + 1) \), defined as

\[
f(t + k) = a_2(t + k)^2 + a_1(t + k) + a_0
\]

\[
= a_2k^2 + (2ta_2 + a_1)k + (t^2a_2 + ta_1 + a_0)
\]

\[
= u_2(t)k^2 + u_1(t)k + u_0(t)
\]

by letting

\[
u_2(t) = a_2
\]

\[
u_1(t) = 2ta_2 + a_1
\]

\[
u_0(t) = t^2a_2 + ta_1 + a_0
\]
(we’ll omit the index \( t \) and write \( u_2, u_1, \) and \( u_0 \).) Then \( \hat{z}(t + 1) = u_2 + u_1 + u_0 \) and

\[
\begin{bmatrix}
  f(t) \\
  f(t-1) \\
  \vdots \\
  f(t-9)
\end{bmatrix}
= \begin{bmatrix}
  0^2 & 0 & 1 \\
  (-1)^2 & -1 & 1 \\
  \vdots & \vdots & \vdots \\
  (-9)^2 & -9 & 1
\end{bmatrix}
\begin{bmatrix}
  u_2 \\
  u_1 \\
  u_0
\end{bmatrix}.
\]

Since the mapping between \((u_2, u_1, u_0)\) and \((a_2, a_1, a_0)\) is one-to-one, we may minimize \( \sum_{\tau=t-9}^t (z(\tau) - f(\tau))^2 \) respect to \((u_2, u_1, u_0)\) instead of respect to \((a_2, a_1, a_0)\). This is a least-squares problem since \( A \) is skinny and full rank. The optimal solution is \( u_{ls} = (A^T A)^{-1} A^T y \), where \( y = (z(t), z(t-1), \ldots, z(t-9)) \). The prediction is given by

\[ \hat{z}(t + 1) = [1 \ 1 \ 1] u_{ls} = [1 \ 1 \ 1] (A^T A)^{-1} A^T y \]

Therefore \( c = A(A^T A)^{-1} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \). So \( c \) is indeed independent of \( n \) and is equal to:

\[
c = \begin{bmatrix}
  0.900 \\
  0.500 \\
  0.183 \\
  -0.050 \\
  -0.200 \\
  -0.267 \\
  -0.250 \\
  -0.150 \\
  0.033 \\
  0.300
\end{bmatrix}^T
\]

Observe that \( c \) does not depend on \( t \), but the coefficients \( a_0, a_1 \) and \( a_2 \) do. In other words, the quadratic extrapolator \( f \) varies between samples, but its value at \( t + 1 \) is always given by the same combination of the previous samples.

(b) The matlab code shown in part (c) computes the predicted values and finds that the relative RMS error is 0.051.

(c) The following matlab code plots the first 100 samples of the prediction.

```matlab
% t = 1:1000; % z = 5*sin(t/10 + 2) + 0.1*sin(t) + 0.1*sin(2*t - 5); A = [(0:-1:-9).^2 ; 0:-1:-9 ; ones(1,10)]'; c = [1 1 1]*pinv(A); % for j=11:1000 % ls_quad_prediction(j) = c*z(j-1:-1:j-10)'; end % residual_ls = ls_quad_prediction(11:1000) - z(11:1000); % rel_rms_ls = sqrt(mean(residual_ls.^2)/mean(z(4:end).^2)) % figure; plot(t, z, '-' , t, ls_quad_prediction, '--'); xlim([0, 100]) xlabel('t') ylabel('value') legend('z(t)', 'zls(t)');
```
7. **Householder reflections.** A *Householder matrix* is defined as

\[ Q = I - 2uu^T, \]

where \( u \in \mathbb{R}^n \) is normalized, that is, \( u^T u = 1 \).

a) Show that \( Q \) is orthogonal.

b) Show that \( Qu = -u \). Show that \( Qv = v \), for any \( v \) such that \( u^Tv = 0 \). Thus, multiplication by \( Q \) gives reflection through the plane with normal vector \( u \).

c) Show that \( \det Q = -1 \).
d) Given a vector $x \in \mathbb{R}^n$, find a unit-length vector $u$ for which $Qx$ lies on the line through $e_1$. Hint: Try a $u$ of the form $u = v/\|v\|$, with $v = x + \alpha e_1$ (find the appropriate $\alpha$ and show that such a $u$ works . . . ) Compute such a $u$ for $x = (3, 2, 4, 1, 5)$. Apply the corresponding Householder reflection to $x$ to find $Qx$.

Note: Multiplication by an orthogonal matrix has very good numerical properties, in the sense that it does not accumulate much roundoff error. For this reason, Householder reflections are used as building blocks for fast, numerically sound algorithms.

Solution.

a) 

$$Q^TQ = (I - 2uu^T)(I - 2uu^T)$$

$$= (I - 2uu^T)(I - uu^T)$$

$$= I - 2uu^T - 2uu^T + 4uu^T uu^T$$

$$= I - 2uu^T - 2uu^T + 4uu^T$$

using $u^T u = 1$

$$= I$$

so $Q$ is orthogonal

b) 

$$Qu = u - 2uu^T u = u - 2u = -u$$

using $u^T u = 1$

$$Qv = v - 2uu^T v = v$$

using $u^T v = 0$

c) We know $\det(Q) = \prod_{i=1}^{n} \lambda_i$. Since $Q$ is symmetric, all eigenvalues are real and we can construct an orthonormal eigenvector basis. From parts (a) and (b), $u$ is an eigenvector with associated eigenvalue $-1$, and any vector $v$ orthogonal to $u$ is an eigenvector with associated eigenvalue $1$. The nullspace of $u^T$ has dimension $n - 1$, so we can construct an orthogonal eigenbasis with all eigenvalues $1$ except for the $-1$ eigenvalue with eigenvector $u$. Thus the product of the eigenvalues is $-1 = \det(Q)$. Some people used a geometric argument, saying that “since $Q$ is a reflection matrix, $\det Q = 0$”. This is OK, as long as valid arguments are provided for $Q$ being a reflection matrix!

d) We follow the hint and choose $u = (x + \alpha e_1)/\|x + \alpha e_1\|$. Then

$$Q = I - 2\frac{(x + \alpha e_1)(x + \alpha e_1)^T}{(x + \alpha e_1)^T(x + \alpha e_1)}$$

$$= I - 2\frac{x(x^T + \alpha e_1^T) + \alpha e_1(x^T + \alpha e_1^T)}{x^T x + \alpha e_1^T x + \alpha x^T e_1 + \alpha^2 e_1^T e_1}$$

$$Qx = x - 2\frac{x(\|x\|^2 + \alpha e_1^T x) + e_1(\|x\|^2 + \alpha^2 x_1)}{\|x\|^2 + 2\alpha x_1 + \alpha^2}$$

$$= x - \frac{2\|x\|^2 + 2\alpha x_1}{\|x\|^2 + 2\alpha x_1 + \alpha^2} x - 2\alpha \frac{\|x\|^2 + \alpha x_1}{\|x\|^2 + 2\alpha x_1 + \alpha^2} e_1$$

$$= \left(1 - \frac{2\|x\|^2 + 2\alpha x_1}{\|x\|^2 + 2\alpha x_1 + \alpha^2}\right) x - 2\alpha \frac{\|x\|^2 + \alpha x_1}{\|x\|^2 + 2\alpha x_1 + \alpha^2} e_1$$

Need this zero
We can achieve this by choosing $\alpha = \pm \|x\|$. This leads to $Qx = \mp \|x\|e_1$ (which makes sense ... $Q$ should always preserve norm). Some people used a geometric argument here as well, and this can make the solution a lot neater if it’s well presented. The idea is to find a reflection plane that reflects the given vector onto the $e_1$ axis (there are two possibilities, for negative and positive parts of the $e_1$ axis), and $u$ is then a unit vector orthogonal to this plane.

8. 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 $x_i \in \mathbb{R}^{n_i}$ to the $i$th receiver, without interference from the other transmitters. All receivers have access to the same signal $y \in \mathbb{R}^m$, which includes the signals of all transmitters, according to

$$y = A_1x_1 + \cdots + A_kx_k,$$

where $A_i \in \mathbb{R}^{m \times n_i}$. 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 $i$th signal is decodable if the $i$th 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 $i$th transmitted signal, while rejecting any interference from the other signals $x_1, x_{i-1}, x_{i+1}, \ldots, x_k$. Whether or not the $i$th 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. $\text{rank}(A_1) < n_1$.

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

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

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

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 \( \text{rank}(A_i) = n_i \).

Now let’s see what happens when the others start transmitting. Suppose \( \text{range}(A_i) \cap \text{range}([A_1 \cdots A_{i-1} A_{i+1} \cdots A_k]) \neq \{0\} \).

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

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

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 \( \text{range}(A_i) \cap \text{range}(Q) = \{0\} \). 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 \( \text{range}(A_i) \), is equal to \( -Qv \), which is in \( \text{range}(Q) \). 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

\[
B_1 A_i = I, \quad B_1 Q = 0.
\]

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: \(\text{rank}(A_1) = n_1\) and
\[
\text{range}(A_i) \cap \text{range}([A_1 \cdots A_{i-1} A_{i+1} \cdots A_k]) = \{0\}.
\]
This last condition is equivalent to
\[
\text{rank}([A_1 \cdots A_k]) = \text{rank}(A_1) + \text{rank}([A_2 \cdots A_k]).
\]
Combining the two conditions, we get
\[
\text{rank}([A_1 \cdots A_k]) = n_1 + \text{rank}([A_2 \cdots A_k]),
\]
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, \textit{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
\[
\text{rank}([A_1 \cdots A_k]) = n_1 + \cdots + n_k,
\]
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
\[
\text{rank}([A_2 \cdots A_k]) = n_2 + \cdots + n_k,
\]
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
\[
\text{range}(A_1) \cap \text{range}([A_2 \cdots A_k]).
\]
If this subspace were nonzero, then a nonzero signal from transmitter 1 produces exactly the same signal as some 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 \( i = 2, \ldots, k \).