

## EE263 Homework 3

Fall 2025

**2.50. Some linear functions associated with a convolution system.** Suppose that  $u$  and  $y$  are scalar-valued discrete-time signals (*i.e.*, sequences) related via convolution:

$$y(k) = \sum_j h_j u(k-j), \quad k \in \mathbb{Z},$$

where  $h_k \in \mathbb{R}$ . You can assume that the convolution is *causal*, *i.e.*,  $h_j = 0$  when  $j < 0$ .

a) *The input/output (Toeplitz) matrix.* Assume that  $u(k) = 0$  for  $k < 0$ , and define

$$U = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(N) \end{bmatrix}, \quad Y = \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N) \end{bmatrix}.$$

Thus  $U$  and  $Y$  are vectors that give the first  $N + 1$  values of the input and output signals, respectively. Find the matrix  $T$  such that  $Y = TU$ . The matrix  $T$  describes the linear mapping from (a chunk of) the input to (a chunk of) the output.  $T$  is called the input/output or Toeplitz matrix (of size  $N + 1$ ) associated with the convolution system.

b) *The Hankel matrix.* Now assume that  $u(k) = 0$  for  $k > 0$  or  $k < -N$  and let

$$U = \begin{bmatrix} u(0) \\ u(-1) \\ \vdots \\ u(-N) \end{bmatrix}, \quad Y = \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N) \end{bmatrix}.$$

Here  $U$  gives the *past input* to the system, and  $Y$  gives (a chunk of) the resulting future output. Find the matrix  $H$  such that  $Y = HU$ .  $H$  is called the Hankel matrix (of size  $N + 1$ ) associated with the convolution system.

### Solution.

a) *The input/output (Toeplitz) matrix.* Since  $h_j = 0$  for  $j < 0$  and  $u(k) = 0$  for  $k < 0$  we have

$$y(k) = \sum_{j=0}^k h_j u(k-j) = \sum_{j=0}^k h_{k-j} u(j),$$

so for  $k = 0, 1, 2, \dots, N$ ,

$$\begin{aligned} y(0) &= h_0 u(0) \\ y(1) &= h_1 u(0) + h_0 u(1) \\ y(2) &= h_2 u(0) + h_1 u(1) + h_0 u(2) \\ &\vdots \\ y(N) &= h_N u(0) + h_{N-1} u(1) + h_{N-2} u(2) + \dots + h_0 u(N) \end{aligned}$$

These equations can be written in matrix form as

$$\underbrace{\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix}}_Y = \underbrace{\begin{bmatrix} h_0 & 0 & 0 & \cdots & 0 \\ h_1 & h_0 & 0 & \cdots & 0 \\ h_2 & h_1 & h_0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \\ h_N & h_{N-1} & \cdots & h_1 & h_0 \end{bmatrix}}_T \underbrace{\begin{bmatrix} u(0) \\ u(1) \\ u(2) \\ \vdots \\ u(N) \end{bmatrix}}_U$$

(Note: the matrix  $T$  above is such that its  $(i, j)$ th entry depends only on the value of  $i - j$ . In other words, it is “constant along the diagonals.” Such matrices are called Toeplitz matrices.)

b) *The Hankel matrix.* Since  $u(k) = 0$  for  $k > 0$  or  $k < -N$

$$y(k) = \sum_{j=k}^{N+k} h_j u(k-j)$$

so for  $k = 0, 1, 2, \dots, N$

$$\begin{aligned} y(0) &= h_0 u(0) + h_1 u(-1) + \cdots + h_N u(-N) \\ y(1) &= h_1 u(0) + h_2 u(-1) + \cdots + h_{N+1} u(-N) \\ y(2) &= h_2 u(0) + h_3 u(-1) + \cdots + h_{N+2} u(-N) \\ &\vdots \\ y(N) &= h_N u(0) + h_{N+1} u(-1) + \cdots + h_{N+N} u(-N) \end{aligned}$$

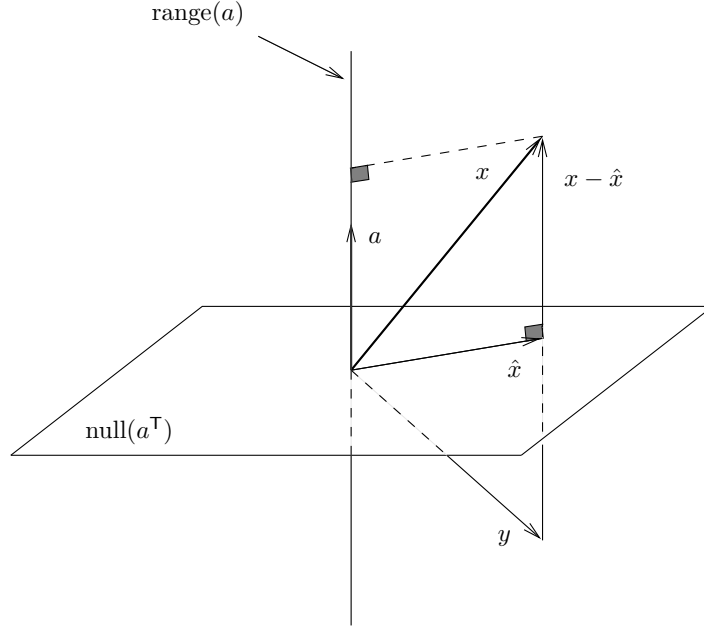
and therefore

$$\underbrace{\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix}}_Y = \underbrace{\begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_N \\ h_1 & h_2 & h_3 & \cdots & h_{N+1} \\ h_2 & h_3 & h_4 & \cdots & h_{N+2} \\ \vdots & & & & \vdots \\ h_N & h_{N+1} & h_{N+2} & \cdots & h_{N+N} \end{bmatrix}}_H \underbrace{\begin{bmatrix} u(0) \\ u(-1) \\ u(-2) \\ \vdots \\ u(-N) \end{bmatrix}}_U$$

(Note: matrices such as  $H$  above that are “constant along the antidiagonals” are called Hankel matrices.)

**4.590. Reflection through a hyperplane.** Find the matrix  $R \in \mathbb{R}^{n \times n}$  such that reflection of  $x$  through the hyperplane  $\{z \mid a^\top z = 0\}$  (with  $a \neq 0$ ) is given by  $Rx$ . Verify that the matrix  $R$  is orthogonal. (To reflect  $x$  through the hyperplane means the following: find the point  $z$  on the hyperplane closest to  $x$ . Starting from  $x$ , go in the direction  $z - x$  through the hyperplane to a point on the opposite side, which has the same distance to  $z$  as  $x$  does.)

**Solution.** From our intuition in three dimensions we know that the reflection of a point through a plane is a linear transformation, *i.e.*, the reflection of a linear combination of vectors is equal to the same linear combination of the reflected vectors. Moreover, this linear transformation is orthogonal because it preserves the length of a vector and the angle between two vectors. We will show that these facts are also true in higher dimensions. More explicitly, given an arbitrary point  $x \in \mathbb{R}^n$ , we will find an orthogonal matrix  $R \in \mathbb{R}^{n \times n}$  such that  $y = Rx$  is the reflection of  $x$  through the hyperplane  $\{z \mid a^\top z = 0\}$ .



By definition, if  $y$  is the reflection of  $x$  we have  $y = x - 2(x - \hat{x})$  where  $\hat{x}$  is the projection of  $x$  on the hyperplane  $\{z \mid a^\top z = 0\}$ . Therefore, it suffices to compute  $\hat{x}$  or  $(x - \hat{x})$  in terms of  $x$  to find the explicit relationship between  $y$  and  $x$ . The vector  $\hat{x}$  is the projection of  $x$  on  $\mathcal{N}(a^\top) = \{z \mid a^\top z = 0\}$ , and therefore,  $x - \hat{x} \perp \mathcal{N}(a^\top)$ . But from a basic fact from linear algebra we know that  $\text{null}(a^\top)$  and  $\text{range}(a)$  are complementary subspaces (they are orthogonal and together they span  $\mathbb{R}^n$ ) so  $x - \hat{x} \in \text{range}(a)$ . In other words,  $\hat{x}$  is the component of  $x$  in  $\text{null}(a^\top)$ , and  $x - \hat{x}$  is the component of  $x$  in  $\text{range}(a)$ . Therefore,  $x - \hat{x}$  can be found by projecting  $x$  on  $\text{range}(a)$ , and since an orthonormal basis for  $\text{range}(a)$  is simply  $\{a/\|a\|\}$  we get

$$\begin{aligned} x - \hat{x} &= \left\langle \frac{a}{\|a\|}, x \right\rangle \frac{a}{\|a\|} \\ &= \frac{a^\top x}{\|a\|^2} a. \end{aligned}$$

As a result

$$\begin{aligned} y &= x - 2(x - \hat{x}) \\ &= x - 2 \frac{a^\top x}{\|a\|^2} a, \end{aligned}$$

so we have expressed  $y$  in terms of  $x$  and we are almost done. To find the matrix  $R$  such that  $y = Rx$  we need to factor  $x$  from the right in  $x - 2\frac{a^\top x}{\|a\|^2}a$ . Since  $\frac{a^\top x}{\|a\|^2}$  is a scalar we have

$$\begin{aligned} y &= x - 2a\frac{a^\top x}{\|a\|^2} \\ &= x - 2\frac{aa^\top}{\|a\|^2}x \\ &= (I - 2\frac{aa^\top}{\|a\|^2})x \end{aligned}$$

and therefore

$$R = I - 2\frac{aa^\top}{\|a\|^2}.$$

Finally, we verify that  $R$  is orthogonal:

$$\begin{aligned} R^\top R &= \left(I - 2\frac{aa^\top}{\|a\|^2}\right)^\top \left(I - 2\frac{aa^\top}{\|a\|^2}\right) \\ &= I - 4\frac{aa^\top}{\|a\|^2} + 4\frac{aa^\top aa^\top}{\|a\|^4} \\ &= I - 4\frac{aa^\top}{\|a\|^2} + 4\frac{aa^\top(a^\top a)}{\|a\|^4} && \text{(since } a^\top a \text{ is a scalar)} \\ &= I - 4\frac{aa^\top}{\|a\|^2} + 4\frac{aa^\top}{\|a\|^2} \\ &= I. \end{aligned}$$

(Note: this problem can also be solved by first computing  $\hat{x}$  from the optimization problem (variable is  $\hat{x}$  not  $x$ )

$$\begin{aligned} &\text{minimize} && \|\hat{x} - x\|^2 \\ &\text{subject to:} && a^\top \hat{x} = 0 \end{aligned}$$

(this can be done by projection arguments or Lagrange multipliers). Then, now that  $\hat{x}$  is known in terms of  $x$  and  $a$ , we can substitute for it in  $y = x - 2(x - \hat{x})$  to get  $R$ .)

**4.600. 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 Julia functions `nullspace` 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 Julia, we can compute a basis for the nullspace of  $A^T$  using the command `null`. (In fact `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 Julia code:

```
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

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

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 Julia via

```
[Q,R]=qr(A);
Q2 = Q[:,[3,4,5]]; # get the last three columns of Q
B = Q2';
B*A
rank(B)
```

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.

**4.630. Groups of equivalent statements.** In the list below there are 11 statements about two square matrices  $A$  and  $B$  in  $\mathbb{R}^{n \times n}$ .

- $\text{range}(B) \subseteq \text{range}(A)$ .
- there exists a matrix  $Y \in \mathbb{R}^{n \times n}$  such that  $B = YA$ .
- $AB = 0$ .
- $BA = 0$ .
- $\text{rank}(\begin{bmatrix} A & B \end{bmatrix}) = \text{rank}(A)$ .

- f)  $\text{range}(A) \perp \text{null}(B^T)$ .
- g)  $\text{rank}\left(\begin{bmatrix} A \\ B \end{bmatrix}\right) = \text{rank}(A)$ .
- h)  $\text{range}(A) \subseteq \text{null}(B)$ .
- i) there exists a matrix  $Z \in \mathbb{R}^{n \times n}$  such that  $B = AZ$ .
- j)  $\text{rank}\left(\begin{bmatrix} A & B \end{bmatrix}\right) = \text{rank}(B)$ .
- k)  $\text{null}(A) \subseteq \text{null}(B)$ .

Your job is to collect them into (the largest possible) groups of equivalent statements. Two statements are equivalent if each one implies the other. For example, the statement ‘ $A$  is onto’ is equivalent to ‘ $\text{null}(A) = \{0\}$ ’ (when  $A$  is square, which we assume here), because every square matrix that is onto has zero nullspace, and vice versa. Two statements are not equivalent if there exist (real) square matrices  $A$  and  $B$  for which one holds, but the other does not. A group of statements is equivalent if any pair of statements in the group is equivalent.

We want *just* your answer, which will consist of lists of mutually equivalent statements; we do not need any justification.

Put your answer in the following specific form. List each group of equivalent statements on a line, in (alphabetic) order. Each new line should start with the first letter not listed above. For example, you might give your answer as

a, c, d, h  
 b, i  
 e  
 f, g, j, k.

This means you believe that statements a, c, d, and h are equivalent; statements b and i are equivalent; and statements f, g, j, and k are equivalent. You also believe that the first group of statements is not equivalent to the second, or the third, and so on.

**Solution.** Let  $b_i$  be the  $i$ th column of  $B$ .

$$\begin{aligned}
 \text{range}(B) \subseteq \text{range}(A) &\Leftrightarrow \text{every column of } B \text{ is in the range of } A \\
 &\Leftrightarrow \text{there exists a vector } z_i \text{ such that } b_i = Az_i \\
 &\Leftrightarrow \text{there exists a matrix } Z \in \mathbb{R}^{n \times n} \text{ such that } B = AZ \\
 &\Leftrightarrow \text{rank}\left(\begin{bmatrix} A & B \end{bmatrix}\right) = \text{rank}(A). \tag{1}
 \end{aligned}$$

This shows that statements a, e and i are equivalent.

$$\begin{aligned}
\text{null}(A) \subseteq \text{null}(B) &\Leftrightarrow \text{null}(A)^\perp \supseteq \text{null}(B)^\perp \\
&\Leftrightarrow \text{range}(B^\top) \subseteq \text{range}(A^\top) \\
&\Leftrightarrow \text{there exists a matrix } \tilde{Y} \in \mathbb{R}^{n \times n} \text{ such that } B^\top = A^\top \tilde{Y} \\
&\Leftrightarrow \text{there exists a matrix } Y \in \mathbb{R}^{n \times n} \text{ such that } B = YA \\
&\Leftrightarrow \text{rank}(\begin{bmatrix} A^\top & B^\top \end{bmatrix}) = \text{rank}(A^\top) \\
&\Leftrightarrow \text{rank}\left(\begin{bmatrix} A \\ B \end{bmatrix}\right) = \text{rank}(A). \tag{2}
\end{aligned}$$

This shows that statements b, g and k are equivalent.

$$\begin{aligned}
\text{range}(A) \subseteq \text{null}(B) &\Leftrightarrow \text{for all } z \in \mathbb{R}^n, B(Az) = 0 \\
&\Leftrightarrow BA = 0. \tag{3}
\end{aligned}$$

This shows that statements d and h are equivalent.

$$\begin{aligned}
\text{range}(A) \perp \text{null}(B^\top) &\Leftrightarrow \text{range}(A) \subseteq \text{null}(B^\top)^\perp \\
&\Leftrightarrow \text{range}(A) \subseteq \text{range}(B) \\
&\Leftrightarrow \text{rank}(\begin{bmatrix} A & B \end{bmatrix}) = \text{rank}(B). \tag{4}
\end{aligned}$$

This shows that statements f and j are equivalent.

None of these groups of statements is equivalent to any other, or to c. This is demonstrated by the following counterexamples.

Take

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}.$$

Since  $AB = 0$  but  $BA \neq 0$ , then group (3) and statement c are not equivalent. Furthermore since

$$\text{rank}\left(\begin{bmatrix} A \\ B \end{bmatrix}\right) = \text{rank}(A) = \text{rank}(B) = 1$$

but  $\text{rank}(\begin{bmatrix} A & B \end{bmatrix}) = 2$ , groups (2) and (1) are not equivalent. Groups (2) and (4) are not either.

When  $A = B \neq 0$ ,  $\text{null}(A) = \text{null}(B)$  but  $AB = BA = A^2 \neq 0$ . Hence groups (2) and (3) are not equivalent. Group (2) and statement c are not equivalent either.

Take

$$A = I, \quad B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}.$$

Since  $\text{rank}(\begin{bmatrix} A & B \end{bmatrix}) = \text{rank}(A) = 2$  but  $\text{rank}(B) = 1$ , groups (1) and (4) are not equivalent. Furthermore since  $BA \neq 0$  groups (1) and (3) are not equivalent. Since  $AB \neq 0$ , group (1) and statement c aren't either.

In a similar fashion, taking

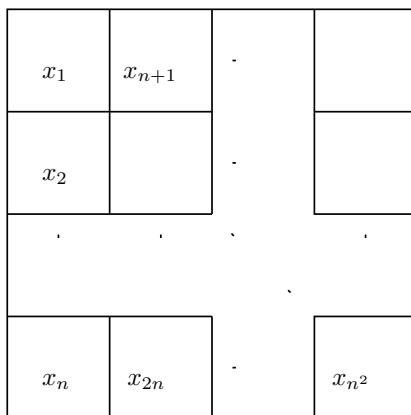
$$A = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad B = I,$$

shows that groups (3) and (4) are not equivalent and that statement c and group (4) aren't either.

Thus, the final answer is

- a, e, i
- b, g, k
- c
- d, h
- f, j.

**6.741. Image reconstruction from line integrals.** In this problem we explore a simple version of a tomography problem. We consider a square region, which we divide into an  $n \times n$  array of square pixels, as shown below.

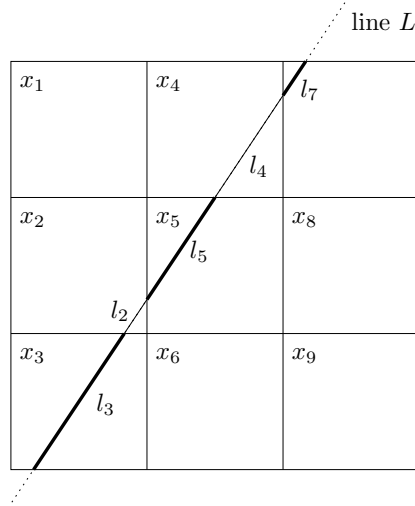


The pixels are indexed column first, by a single index  $i$  ranging from 1 to  $n^2$ , as shown above. We are interested in some physical property such as density (say) which varies over the region. To simplify things, we'll assume that the density is constant inside each pixel, and we denote by  $x_i$  the density in pixel  $i$ ,  $i = 1, \dots, n^2$ . Thus,  $x \in \mathbb{R}^{n^2}$  is a vector that describes the density across the rectangular array of pixels. The problem is to estimate the vector of densities  $x$ , from a set of sensor measurements that we now describe. Each sensor measurement is a *line integral* of the density over a line  $L$ . In addition, each measurement is corrupted by a (small) noise term. In other words, the sensor measurement for line  $L$  is given by

$$\sum_{i=1}^{n^2} l_i x_i + v,$$

where  $l_i$  is the length of the intersection of line  $L$  with pixel  $i$  (or zero if they don't intersect), and  $v$  is a (small) measurement noise. This is illustrated below for a problem with  $n = 3$ . In

this example, we have  $l_1 = l_6 = l_8 = l_9 = 0$ .



Now suppose we have  $N$  line integral measurements, associated with lines  $L_1, \dots, L_N$ . From these measurements, we want to estimate the vector of densities  $x$ . The lines are characterized by the intersection lengths

$$l_{ij}, \quad i = 1, \dots, n^2, \quad j = 1, \dots, N,$$

where  $l_{ij}$  gives the length of the intersection of line  $L_j$  with pixel  $i$ . Then, the whole set of measurements forms a vector  $y \in \mathbb{R}^N$  whose elements are given by

$$y_j = \sum_{i=1}^{n^2} l_{ij} x_i + v_j, \quad j = 1, \dots, N.$$

And now the problem: you will reconstruct the pixel densities  $x$  from the line integral measurements  $y$ . The class webpage contains the file `tomo_data.json`, which contains the following variables:

- `N`, the number of measurements ( $N$ ),
- `npixels`, the side length in pixels of the square region ( $n$ ),
- `y`, a vector with the line integrals  $y_j$ ,  $j = 1, \dots, N$ ,
- `line_pixel_lengths`, an  $n^2 \times N$  matrix containing the intersection lengths  $l_{ij}$  of each pixel  $i = 1, \dots, n^2$  (ordered column-first as in the above diagram) and each line  $j = 1, \dots, N$ ,
- `lines_d`, a vector containing the displacement (distance from the center of the region in pixel lengths)  $d_j$  of each line  $j = 1, \dots, N$ , and
- `lines_theta`, a vector containing the angles  $\theta_j$  of each line  $j = 1, \dots, N$ .

(You shouldn't need `lines_d` or `lines_theta`, but we're providing them to give you some idea of how the data was generated. Similarly, the file `tmeasure.jl` shows how we computed the measurements, but you don't need it or anything in it to solve the problem. The variable `line_pixel_lengths` was computed using the function in this file.)

Use this information to find  $x$ , and display it as an image (of  $n$  by  $n$  pixels). You'll know you have it right.

*Julia hints:*

- The `reshape` function might help with converting between vectors and matrices, for example, `A = reshape(v, m, n)` will convert a vector with  $v = mn$  elements into an  $m \times n$  matrix.
- To display a matrix `A` as a grayscale image, you can use: (or any method that works for you)  

```
heatmap(A, yflip=true, aspect_ratio=:equal, color=:gist_gray,
        cbar=:none, framestyle=:none)
```

You'll need to have loaded the JuliaPlots package with `using Plots` to access the `heatmap` function. (The `yflip` argument gets it to plot the origin in the top-left rather than the bottom-left.)

*Note:* While irrelevant to your solution, this is actually a simple version of *tomography*, best known for its application in medical imaging as the CAT scan. If an *x-ray* gets attenuated at rate  $x_i$  in pixel  $i$  (a little piece of a cross-section of your body), the  $j$ -th measurement is

$$z_j = \prod_{i=1}^{n^2} e^{-x_i l_{ij}},$$

with the  $l_{ij}$  as before. Now define  $y_j = -\log z_j$ , and we get

$$y_j = \sum_{i=1}^{n^2} x_i l_{ij}.$$

**Solution.** The first thing to do is to restate the problem in the familiar form  $y = Ax + v$ . Here,  $y \in \mathbb{R}^N$  is the measurement (given),  $x \in \mathbb{R}^{n^2}$  is the physical quantity we are interested in,  $A \in \mathbb{R}^{N \times n^2}$  is the relation between them, and  $v \in \mathbb{R}^N$  is the noise, or measurement error (unknown, and we'll not worry about it). So we need to find the elements of  $A$  ... how do we do that?

Comparing  $y = Ax$ , *i.e.*,  $y_j = \sum_{i=1}^{n^2} A_{ji} x_i$  with our model  $y_j = \sum_{i=1}^{n^2} l_{ij} x_i + v_j$

it is clear that  $A_{ji} = l_{ij}$ ,  $j = 1 \dots N$ ,  $i = 1 \dots n^2$ . We have thus determined a standard linear model that we want to "invert" to find  $x$ . On running `tomo_data.json`, we find that  $n = 30$  and  $N = 1225$ , so  $N > n^2$ , *i.e.*, we have more rows than columns – a skinny matrix. If  $A$  is full-rank the problem is overdetermined. We can find a unique (but not exact) solution  $x_{ls}$  – the least-squares solution that minimizes  $\|Ax - y\|$  – which, due to its noise-reducing

properties, provides a good estimate of  $x$  (it is the *best linear unbiased estimate*). So here's what we do: we construct  $A$  element for element by finding the length of the intersection of each line with each pixel. That's done using `line_pixel_length.jl` provided.  $A$  turns out to be full-rank (`rank(A)` returns 64), so we can compute a unique  $x_{ls}$ . We go ahead and solve the least-squares problem, and then display the result.

Here comes a translation of the above paragraph into Julia code:

```
using LinearAlgebra
using Plots

include("readclassjson.jl")
data = readclassjson("tomo_data.json")
N = data["N"]
L = data["line_pixel_lengths"]
npixels = data["npixels"]
y = data["y"];

x = (L*L') \ L * y
image = reshape(x, npixels, npixels)
heatmap(image, yflip=true, aspect_ratio=:equal, \
        color=:gist_gray, cbar=:none, framestyle=:none)
```

And here's the end result, the reconstructed image

