

EE263 Homework 3
Fall 2025
due Thursday 10/16, at 11:59 PM

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.

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.)

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.

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 $B = YA$.
- $AB = 0$.
- $BA = 0$.
- $\text{rank}(\begin{bmatrix} A & B \end{bmatrix}) = \text{rank}(A)$.
- $\text{range}(A) \perp \text{null}(B^T)$.
- $\text{rank}\left(\begin{bmatrix} A \\ B \end{bmatrix}\right) = \text{rank}(A)$.
- $\text{range}(A) \subseteq \text{null}(B)$.

- i) there exists a matrix $Z \in \mathbb{R}^{n \times n}$ such that $B = AZ$.
- j) $\text{rank}([A \ B]) = \text{rank}(B)$.
- 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.

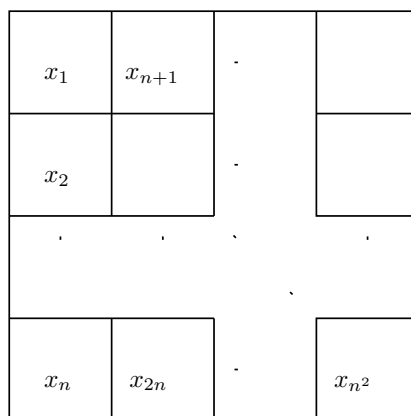
We want *just* your answer, which will consist of lists of mutually equivalent statements; we do not need any justification.

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.

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.

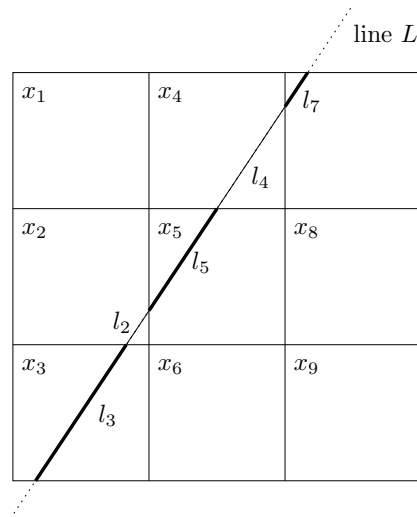


The pixels are indexed column first, by a single index i ranging from 1 to n^2 , as shown above. We are interested in some physical property such as density (say) which varies over the region. To simplify things, we’ll assume that the density is constant inside each pixel, and we denote

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 θ_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

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