EE263 Homework 4 Solutions

Fall 2023

3.530. Checking some range and nullspace conditions. Explain how to determine whether or not the following statements hold:

a) $\text{range}(A) = \text{range}(B)$.

b) $\text{range}(A) \perp \text{range}(B)$.

c) $\text{range}(A) \cap \text{range}(B) = \{0\}$.

d) $\text{range}(C) \subseteq \text{null}(B)$.

The matrices have dimensions $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{m\times p}$, $C \in \mathbb{R}^{p\times m}$.

Your answer can involve standard matrix operations on the matrices above, such as addition, multiplication, transposition, concatenation (*i.e.*, building block matrices), and inversion, as well as a function $\text{rank}(X)$, that gives the rank of a matrix X, and $\det(X)$, which gives the determinant of a (square) matrix X.

For example, you might assert that (a) holds if and only if $\text{rank}([A\ B]) = m$. (This is not correct; it's just an example of what your answer might look like.)

You do not need to give a proof or long justification that your conditions are correct; a short one or two sentence explanation for each statement is fine. Points will be deducted from correct answers that are substantially longer than they need to be, or are confusing (to us).

Solution.

a) $\text{range}(A) = \text{range}(B)$ if and only if $\text{rank}([A\ B]) = \text{rank}(A) = \text{rank}(B)$.

First assume that $\text{range}(A) = \text{range}(B)$ holds. In general we have $\text{range}([A\ B]) = \text{range}(A) + \text{range}(B)$. Since $\text{range}(A) = \text{range}(B)$, we have since $\text{range}([A\ B]) = \text{range}(A) = \text{range}(B)$ (this is equality of three subspaces of $\mathbb{R}^m$). Taking the dimension of these three (identical) subspaces, we conclude $\text{rank}([A\ B]) = \text{rank}(A) = \text{rank}(B)$.

To show the converse, suppose that $\text{rank}([A\ B]) = \text{rank}(A) = \text{rank}(B)$. We always have $\text{range}([A\ B]) \supseteq \text{range}(A)$ (and same for B). From the assumption, these two subspaces have the same dimension, and we conclude they are equal, *i.e.*, $\text{range}([A\ B]) = \text{range}(A)$. The same holds for B, and we conclude that $\text{range}(A) = \text{range}(B)$.

b) $\text{range}(A) \perp \text{range}(B)$ if and only if $A^\mathsf{T}B = 0$.

First note that $\text{range}(A) \perp \text{range}(B)$ means that for any u and any v, we have $(Au) \perp (Bv)$. This is equivalent to saying $(Au)^\mathsf{T}(Bv) = u^\mathsf{T}(A^\mathsf{T}B)v = 0$ for any u and v, which occurs if and only if $A^\mathsf{T}B = 0$. Recall that this last statement follows from the fact that if $u^\mathsf{T}Cv = 0$ for all u and v, then, in particular, $c_{ij} = e_i^\mathsf{T}Ce_j = 0$ for all i, j.

c) $\text{range}(A) \cap \text{range}(B) = \{0\}$ if and only if $\text{rank}([A\ B]) = \text{rank}(A) + \text{rank}(B)$.

It will be easier to see the above equivalence if we have an orthonormal basis for $\text{range}(A)$ and for $\text{range}(B)$. Thus, let Q_A and Q_B be matrices (coming from, say, a QR factorization) whose columns form a basis for $\text{range}(A)$ and $\text{range}(B)$, respectively. Note that

the ranges of A and Q_A are the same, and so are their ranks. The same is true for B and Q_B. Also, note that $\text{rank}([A\ B]) = \text{rank}([Q_A\ Q_B])$.

With this set up, we can proceed. Note that $\text{range}(Q_A) \cap \text{range}(Q_B) = \{0\}$ if and only if $Q_A x = Q_B y$ is only solved when $Q_A x = 0 = Q_B y$. Since the columns of Q_A and Q_B form independent sets, this happens if and only if $x = 0$ and $y = 0$. If we let $z = -y$, we can rewrite this statement as

$$\begin{bmatrix} Q_A & Q_B \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = 0$$

if and only if $x = 0$ and $z = 0$, which is the same as saying that the columns of $[Q_A\ Q_B]$ are linearly independent, or $[Q_A\ Q_B]$ has full rank. That is,

$$\text{rank}([Q_A\ Q_B]) = \text{rank}(Q_A) + \text{rank}(Q_B).$$

(Each of these matrices is full rank, so the rank is just the number of columns.)

Now, we can finally conclude that $\text{range}(A) \cap \text{range}(B) = \text{range}(Q_A) \cap \text{range}(Q_B) = \{0\}$ if and only if

$$\text{rank}([A\ B]) = \text{rank}([Q_A\ Q_B]) = \text{rank}(Q_A) + \text{rank}(Q_B) = \text{rank}(A) + \text{rank}(B).$$

d) $\text{range}(C) \subseteq \text{null}(B)$ if and only if $BC = 0$.

Stating $\text{range}(C) \subseteq \text{null}(B)$ is equivalent to stating that for any u, $Cu \in \text{null}(B)$. This, in turn, is equivalent to $B(Cu) = 0$ for any u, which occurs if and only if $BC = 0$.

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 occured. 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 $\text{range}(C^\mathsf{T})$ are orthogonal complements. It follows that $\text{null}(B)$ and $\text{range}(B^\mathsf{T})$ are orthogonal complements, and so are $\text{range}(A)$ and $\text{null}(A^\mathsf{T})$. We require that $\text{null}(B) = \text{range}(A)$, so this means their orthogonal complements are equal, *i.e.*, $\text{range}(B^\mathsf{T}) = \text{null}(A^\mathsf{T})$. 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 $B = Q_2^\mathsf{T}$. 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.

4.830. True/false questions about linear algebra. Determine whether each of the following statements is true or false. In each case, give either a proof or a counterexample.

a) If Q has orthonormal columns, then $\|Q^\mathsf{T} w\| \leq \|w\|$ for all vectors w.

b) Suppose $A \in \mathbb{R}^{m\times p}$ and $B \in \mathbb{R}^{m\times q}$. If $\text{null}(A) = \{0\}$ and $\text{range}(A) \subset \text{range}(B)$, then $p \leq q$.

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

d) If $\text{rank}\left(\begin{bmatrix} A & B \end{bmatrix}\right) = \text{rank}(A) = \text{rank}(B)$, then $\text{range}(A) = \text{range}(B)$.

e) Suppose $A \in \mathbb{R}^{m\times n}$. Then, $x \in \text{null}(A^\mathsf{T})$ if and only if $x \notin \text{range}(A)$.

f) Suppose A is invertible. Then, AB is not full rank if and only if B is not full rank.

g) If A is not full rank, then there is a nonzero vector x such that $Ax = 0$.

Solution.

a) The statement is true. Suppose $Q \in \mathbb{R}^{m\times n}$. Because the columns of Q are orthonormal, and hence linearly independent, we know that $m \geq n$. If $m = n$, then Q is an orthogonal matrix, so QQ^T, and we have that

$$\|Q^\mathsf{T} w\|^2 = w^\mathsf{T}(QQ^\mathsf{T})w = w^\mathsf{T} w = \|w\|^2.$$

Now consider the case when $m > n$. Let $q_1, \ldots, q_n$ be the columns of Q:

$$Q = \begin{bmatrix} q_1 & \cdots & q_n \end{bmatrix}.$$

Then, we can extend $(q_1, \ldots, q_n)$ to an orthonormal basis $(q_1, \ldots, q_m)$ for $\mathbb{R}^m$. Define the matrix $\tilde{Q} \in \mathbb{R}^{m \times (m-n)}$ such that

$$\tilde{Q} = \begin{bmatrix} q_{m+1} & \cdots & q_n \end{bmatrix}.$$

Then, we have that $\begin{bmatrix} Q & \tilde{Q} \end{bmatrix}$ is an orthogonal matrix, so that

$$\|w\|^2 = \left\| \begin{bmatrix} Q & \tilde{Q} \end{bmatrix}^\mathsf{T} w \right\|^2 = \left\| \begin{bmatrix} Q^\mathsf{T} \\ \tilde{Q}^\mathsf{T} \end{bmatrix} w \right\|^2 = \|Q^\mathsf{T} w\|^2 + \|\tilde{Q}^\mathsf{T} w\|^2 \geq \|Q^\mathsf{T} w\|^2.$$

Combining these results, we see that if the columns of Q are orthonormal, then $\|Q^\mathsf{T} w\| \leq \|w\|$ for all vectors w. (This result is known as Bessel's inequality.)

b) The statement is true. Because $\text{range}(A) \subset \text{range}(B)$, we have that

$$\text{rank}(A) = \dim(\text{range}(A)) \leq \dim(\text{range}(B)) = \text{rank}(B).$$

Since the rank of a matrix is bounded by its number of columns, we have that

$$\text{rank}(B) \leq q.$$

Conservation of dimension tells us that

$$\text{rank}(A) = \dim(\mathbb{R}^p) - \dim(\text{null}(A)) = \dim(\mathbb{R}^p) - \dim(\{0\}) = p - 0 = p.$$

Combining these results, we have that

$$p = \text{rank}(A) \leq \text{rank}(B) \leq q.$$

c) The statement is true. Suppose $y \in \text{null}(AV_2)$. Then, $V_2 y \in \text{null}(A) = \text{range}(V_1)$, so there exists a vector x such that $V_1 x = V_2 y$. Therefore, we have that

$$V_1 x - V_2 y = \begin{bmatrix} V_1 & V_2 \end{bmatrix} \begin{bmatrix} x \\ -y \end{bmatrix} = 0.$$

Because $\begin{bmatrix} V_1 & V_2 \end{bmatrix}$ is invertible, this implies that

$$\begin{bmatrix} x \\ -y \end{bmatrix} = 0,$$

and hence that $y = 0$. Thus, we see that $\text{null}(AV_2) = \{0\}$.

d) The statement is true. It is sufficient to show that if $\text{rank}\big(\begin{bmatrix} A & B \end{bmatrix}\big) = \text{rank}(A)$, then $\text{range}(A) = \text{range}\big(\begin{bmatrix} A & B \end{bmatrix}\big)$. This implies that if $\text{rank}\big(\begin{bmatrix} A & B \end{bmatrix}\big) = \text{rank}(A) = \text{rank}(B)$, then

$$\text{range}(A) = \text{range}\big(\begin{bmatrix} A & B \end{bmatrix}\big) = \text{range}(B).$$

Consider any $y \in \text{range}(A)$. There exists a vector x such that $Ax = y$. Then, we have that

$$\begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} x \\ 0 \end{bmatrix} = Ax = y,$$

so that $y \in \text{range}\big(\begin{bmatrix} A & B \end{bmatrix}\big)$. Thus, we have that $\text{range}(A) \subset \text{range}\big(\begin{bmatrix} A & B \end{bmatrix}\big)$. (Note that this result holds for any matrices A and B.) Now suppose that $\text{rank}(A) = \text{rank}\big(\begin{bmatrix} A & B \end{bmatrix}\big)$. Let $(q_1, \ldots, q_m)$ be a basis for $\text{range}(A)$. Since $\text{range}(A)$ is a subspace of $\text{range}\big(\begin{bmatrix} A & B \end{bmatrix}\big)$, we can extend this basis to a basis $(q_1, \ldots, q_n)$ for $\text{range}\big(\begin{bmatrix} A & B \end{bmatrix}\big)$. However, it must be the case that

$$n = \text{rank}\big(\begin{bmatrix} A & B \end{bmatrix}\big) = \text{rank}(A) = m.$$

Thus, $(q_1, \ldots, q_m)$ is a basis for both $\text{range}(A)$ and $\text{range}(\begin{bmatrix} A & B \end{bmatrix})$. Therefore, for any $y \in \text{range}(\begin{bmatrix} A & B \end{bmatrix})$, there exist scalars $c_1, \ldots, c_m$ such that

$$y = c_1 q_1 + \cdots + c_m q_m.$$

Because $(q_1, \ldots, q_m)$ is also a basis for $\text{range}(A)$, this implies that $y \in \text{range}(A)$. This shows that if $\text{rank}(A) = \text{rank}\big(\begin{bmatrix} A & B \end{bmatrix}\big)$, then $\text{range}\big(\begin{bmatrix} A & B \end{bmatrix}\big) \subset \text{range}(A)$, and thereby completes the proof.

e) The statement is false. In fact, neither direction of the equivalence is true. Consider the matrix

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

For $x = 0$, we have that $x \in \text{null}(A^\mathsf{T})$, and $x \in \text{range}(A)$; for $x = (1, 1)$, we have that $x \notin \text{range}(A)$, and $x \notin \text{null}(A^\mathsf{T})$.

f) The statement is true. We claim that if A is invertible, then $\text{rank}(AB) = \text{rank}(B)$. This implies that AB is full rank if and only if B is full rank. Let $(q_1, \ldots, q_k)$ be a basis for $\text{range}(B)$. We claim that $(Aq_1, \ldots, Aq_k)$ is a basis for AB. Consider any $y \in \text{range}(AB)$. There exists a vector x such that $(AB)x = y$. Note that Bx is a vector in $\text{range}(B)$, so there exist scalars $c_1, \ldots, c_k$ such that

$$Bx = c_1 q_1 + \cdots + c_k q_k.$$

Then, we have that

$$y = (AB)x = A(Bx) = A(c_1 q_1 + \cdots + c_k q_k) = c_1(Aq_1) + \cdots + c_k(Aq_k).$$

This shows that $(Aq_1, \ldots, Aq_k)$ spans $\text{range}(AB)$. (This is true for any matrices A and B.) Suppose there exist scalars $d_1, \ldots, d_k$ such that

$$d_1(Aq_1) + \cdots + d_k(Aq_k) = A(d_1 q_1 + \cdots + d_k q_k) = 0.$$

Since A is invertible, this implies that

$$d_1 q_1 + \cdots + d_k q_k = A^{-1} 0 = 0.$$

Then, because $(q_1, \ldots, q_k)$ is a basis, and hence linearly independent, we have that

$$d_1 = \cdots = d_k = 0.$$

This shows that $(Aq_1, \ldots, Aq_k)$ is linearly independent, and completes the proof that $(Aq_1, \ldots, Aq_k)$ is a basis for range(AB). Since the dimension of a subspace is the number of vectors in a basis for the subspace, we have that

$$\text{rank}(B) = \dim(\text{range}(B)) = k = \dim(\text{range}(AB)) = \text{rank}(AB).$$

g) The statement is true. If A is strictly fat, then there exists a nonzero vector x such that $Ax = 0$ irrespective of whether of A is full rank. Suppose A is skinny (or square). If A is not full rank, then the columns of A must be linearly dependent: that is, there exist scalars $x_1, \ldots, x_n$, at least one of which is nonzero, such that

$$x_1 A_{*1} + \cdots + x_n A_{*n} = 0.$$

In matrix form, this equation says that

$$\begin{bmatrix} A_{*1} & \cdots & A_{*n} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = Ax = 0,$$

where $x \neq 0$. ■

5.680. Least-squares residuals. Suppose A is skinny and full-rank. Let x_{ls} be the least-squares approximate solution of $Ax = y$, and let $y_{\text{ls}} = Ax_{\text{ls}}$. Show that the residual vector $r = y - y_{\text{ls}}$ satisfies

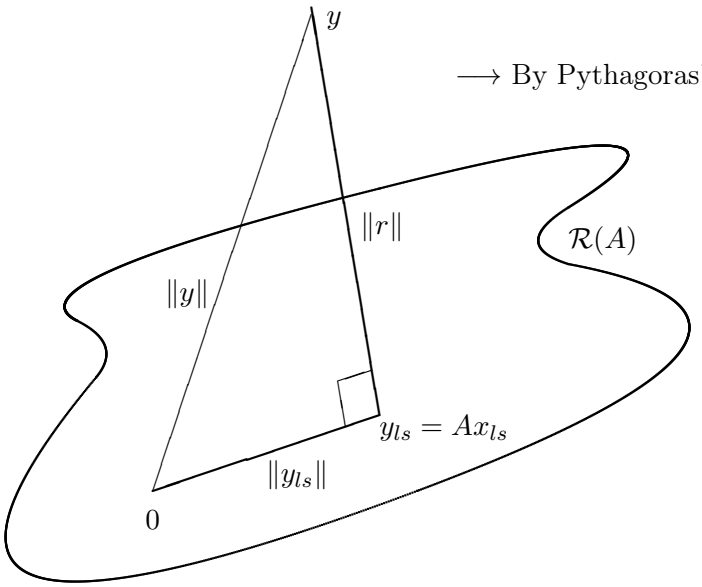
$$\|r\|^2 = \|y\|^2 - \|y_{\text{ls}}\|^2.$$

Also, give a brief geometric interpretation of this equality (just a couple of sentences, and maybe a conceptual drawing).

Solution. Let us first show that $r \perp y_{\text{ls}}$. Since $y_{\text{ls}} = Ax_{\text{ls}} = AA^{\dagger}y = A(A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}}y$

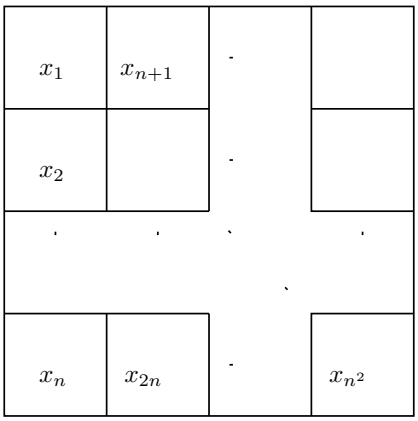
$$\begin{aligned} {y_{\text{ls}}}^{\mathsf{T}} r = {y_{\text{ls}}}^{\mathsf{T}}(y - y_{\text{ls}}) &= {y_{\text{ls}}}^{\mathsf{T}} y \;-\; {y_{\text{ls}}}^{\mathsf{T}} y_{\text{ls}} \\ &= y^{\mathsf{T}} A(A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}} y \;-\; y^{\mathsf{T}} A(A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}}A(A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}} y \\ &= y^{\mathsf{T}} A(A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}} y \;-\; y^{\mathsf{T}} A(A^{\mathsf{T}}A)^{-1}(A^{\mathsf{T}}A)(A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}} y \\ &= y^{\mathsf{T}} A(A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}} y \;-\; y^{\mathsf{T}} A(A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}} y \\ &= 0. \end{aligned}$$

Thus, $\|y\|^2 = \|y_{\text{ls}} + r\|^2 = (y_{\text{ls}} + r)^{\mathsf{T}}(y_{\text{ls}} + r) = \|y_{\text{ls}}\|^2 + 2y_{\text{ls}}{}^{\mathsf{T}} r + \|r\|^2 = \|y_{\text{ls}}\|^2 + \|r\|^2$. Therefore $\|r\|^2 = \|y\|^2 - \|y_{\text{ls}}\|^2$.



$\longrightarrow$ By Pythagoras' theorem, $\|y\|^2 = \|y_{\text{ls}}\|^2 + \|r\|^2$

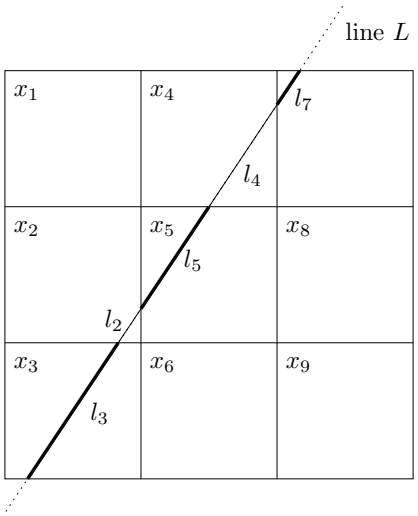
6.741. Image reconstruction from line integrals. In this problem we explore a simple version of a tomography problem. We consider a square region, which we divide into an $n \times n$ array of square pixels, as shown below.



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, $i = 1, \ldots, n^2$. 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 $L_1, \ldots, L_N$. From these measurements, we want to estimate the vector of densities x. The lines are characterized by the intersection lengths

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

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

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

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 $y_j,\ j = 1, \ldots, N$,
- `line_pixel_lengths`, an $n^2 \times N$ matrix containing the intersection lengths l_{ij} of each pixel $i = 1, \ldots, n^2$ (ordered column-first as in the above diagram) and each line $j = 1, \ldots, N$,
- `lines_d`, a vector containing the displacement (distance from the center of the region in pixel lengths) d_j of each line $j = 1, \ldots, N$, and
- `lines_theta`, a vector containing the angles θ_j of each line $j = 1, \ldots, N$.

(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?

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

it is clear that $A_{ji} = l_{ij}, \quad j = 1 \ldots N, \quad i = 1 \ldots n^2$. 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.

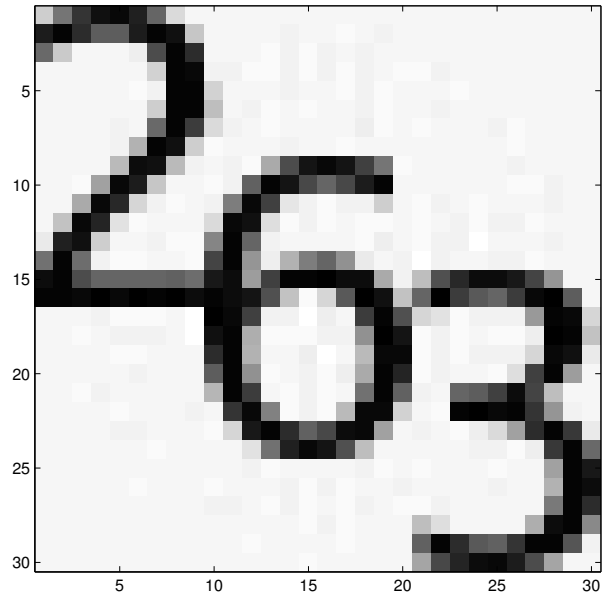
Here comes a translation of the above paragraph into Julia code:

```
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



7.1060. Curve-smoothing. We are given a function $F : [0, 1] \to \mathbb{R}$ (whose graph gives a curve in $\mathbb{R}^2$). Our goal is to find another function $G : [0, 1] \to \mathbb{R}$, which is a *smoothed* version of F. We'll judge the smoothed version G of F in two ways:

- *Mean-square deviation from F,* defined as
$$D = \int_0^1 (F(t) - G(t))^2 \, dt.$$
- *Mean-square curvature,* defined as
$$C = \int_0^1 G''(t)^2 \, dt.$$

We want *both* D and C to be small, so we have a problem with two objectives. In general there will be a trade-off between the two objectives. At one extreme, we can choose $G = F$, which makes $D = 0$; at the other extreme, we can choose G to be an affine function (*i.e.*, to have $G''(t) = 0$ for all $t \in [0, 1]$), in which case $C = 0$. The problem is to identify the optimal trade-off curve between C and D, and explain how to find smoothed functions G on the optimal trade-off curve. To reduce the problem to a finite-dimensional one, we will represent the functions F and G (approximately) by vectors $f,\ g \in \mathbb{R}^n$, where
$$f_i = F(i/n), \quad g_i = G(i/n).$$
You can assume that n is chosen large enough to represent the functions well. Using this representation we will use the following objectives, which approximate the ones defined for the functions above:

- *Mean-square deviation,* defined as
$$d = \frac{1}{n} \sum_{i=1}^{n} (f_i - g_i)^2.$$

- *Mean-square curvature,* defined as

$$c = \frac{1}{n-2}\sum_{i=2}^{n-1}\left(\frac{g_{i+1} - 2g_i + g_{i-1}}{1/n^2}\right)^2.$$

In our definition of c, note that

$$\frac{g_{i+1} - 2g_i + g_{i-1}}{1/n^2}$$

gives a simple approximation of $G''(i/n)$. You will only work with this approximate version of the problem, *i.e.*, the vectors f and g and the objectives c and d.

a) Explain how to find g that minimizes $d + \mu c$, where $\mu \geq 0$ is a parameter that gives the relative weighting of sum-square curvature compared to sum-square deviation. Does your method always work? If there are some assumptions you need to make (say, on rank of some matrix, independence of some vectors, etc.), state them clearly. Explain how to obtain the two extreme cases: $\mu = 0$, which corresponds to minimizing d without regard for c, and also the solution obtained as $\mu \to \infty$ (*i.e.*, as we put more and more weight on minimizing curvature).

b) Get the file `curve_smoothing.json` from the course web site. This file defines a specific vector f that you will use. Find and plot the optimal trade-off curve between d and c. Be sure to identify any critical points (such as, for example, any intersection of the curve with an axis). Plot the optimal g for the two extreme cases $\mu = 0$ and $\mu \to \infty$, and for three values of μ in between (chosen to show the trade-off nicely). On your plots of g, be sure to include also a plot of f, say with dotted line type, for reference.

Solution.

a) Let's start with the two extreme cases. When $\mu = 0$, finding g to minimize $d + \mu c$ reduces to finding g to minimize d. Since d is a sum of squares, $d \geq 0$. Choosing $g = f$ trivially achieves $d = 0$. This makes perfect sense: to minimize the deviation measure, just take the smoothed version to be the same as the original function. This yields zero deviation, naturally, but also, it yields no smoothing! Next, consider the extreme case where $\mu \to \infty$. This means we want to make the curvature as small as possible. Can we drive it to zero? The answer is yes, we can: the curvature is zero if and only if g is an affine function, *i.e.*, has the form $g_i = ai + b$ for some constants a and b. There are lots of vectors g that have this form; in fact, we have one for every pair of numbers a, b. All of these vectors g make c zero. Which one do we choose? Well, even if μ is huge, we still have a small contribution to $d + \mu c$ from d, so among all g that make $c = 0$, we'd like the one that minimizes d. Basically, we want to find the best affine approximation, in the sum of squares sense, to f. We want to find a and b that minimize

$$\left\| f - A\begin{bmatrix} a \\ b \end{bmatrix} \right\| \text{ where } A = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ \vdots & \vdots \\ n & 1 \end{bmatrix}.$$

For $n \geq 2$, A is skinny and full rank, and a and b can be found using least-squares. Specifically, $[a\ b]^\mathsf{T} = (A^\mathsf{T}A)^{-1}A^\mathsf{T}f$. In the general case, minimizing $d + \mu c$, is the same as choosing g to minimize

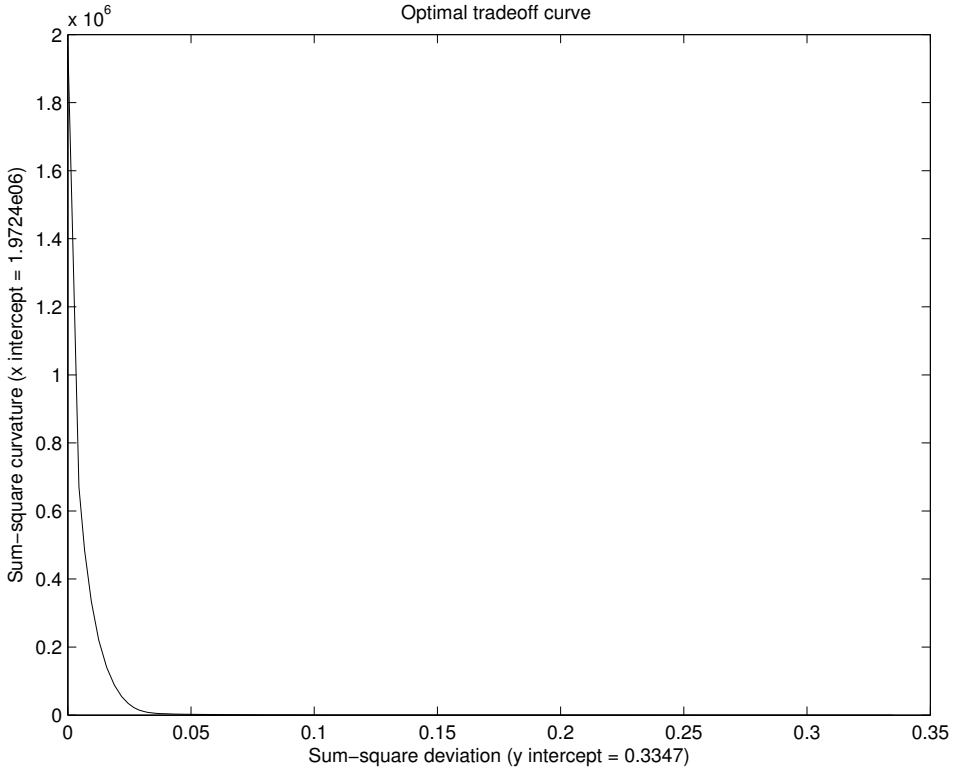
$$\left\| \frac{1}{\sqrt{n}} Ig - \frac{1}{\sqrt{n}} f \right\|^2 + \mu \left\| \underbrace{\frac{n^2}{\sqrt{n-2}} \begin{bmatrix} -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 & 2 & -1 \end{bmatrix}}_{S} g \right\|^2 .$$

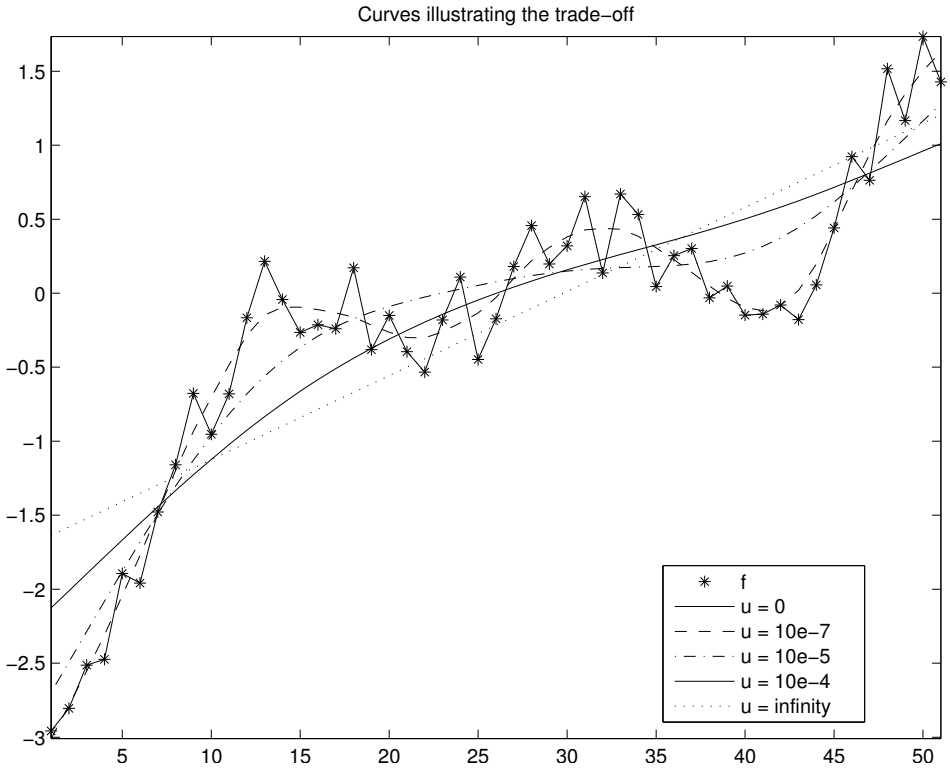
This is a multi-objective least-squares problem. The minimizing g is

$$g = (\tilde{A}^\mathsf{T}\tilde{A})^{-1}\tilde{A}^\mathsf{T}\tilde{y} \text{ where } \tilde{A} = \begin{bmatrix} \frac{I}{\sqrt{n}} \\ \sqrt{\mu}S \end{bmatrix} \text{ and } \tilde{y} = \begin{bmatrix} \frac{f}{\sqrt{n}} \\ 0 \end{bmatrix}.$$

The inverse of $\tilde{A}$ always always exists because I is full rank. The expression can also be written as $g = (\frac{I}{n} + \mu S^\mathsf{T}S)^{-1}\frac{f}{n}$.

b) The following plots show the optimal trade-off curve and the optimal g corresponding to representative μ values on the curve.





The following matlab code finds and plots the optimal trade-off curve between d and c. It also finds and plots the optimal g for representative values of μ. As expected, when $\mu = 0$, $g = f$ and no smoothing occurs. At the other extreme, as μ goes to infinity, we get an affine approximation of f. Intermediate values of μ correspond to approximations of f with different degrees of smoothness.

```
using LinearAlgebra
using Plots
using ToeplitzMatrices

include("readclassjson.jl")
data = readclassjson("curve_smoothing_data.json")
f = data["f"]
n = data["n"]

S = Toeplitz(vec([-1; zeros(n-3,1)]), vec([-1; 2; -1; zeros(n-3,1)]));
S = S*n^2/(sqrt(n-2));
I_n = 1*Matrix(I, n, n)
g_no_deviation = f;

error_curvature = []
error_deviation = []
append!(error_curvature, norm(S*g_no_deviation)^2)
append!(error_deviation, 0)
```

```
u = 10 .^(range(-8,stop=-3,length=30));


for i = eachindex(u)
    A_tilde = [1/sqrt(n)*I_n; sqrt(u[i])*S];
    y_tilde = [1/sqrt(n)*f; zeros(n-2,1)];
    g = A_tilde\y_tilde;
    append!(error_deviation, norm(1/sqrt(n)*I_n*g-f/sqrt(n))^2);
    append!(error_curvature, norm(S*g)^2);
end

a1 = collect(1:n);
a2 = ones(n,1);
A = [a1 a2];
affine_param = inv(A'*A)*A'*f;

g_no_curvature = []
for i = 1:n
    append!(g_no_curvature, affine_param[1]*i+affine_param[2])
end

g_no_curvature = g_no_curvature';
append!(error_deviation, 1/n*norm(vec(g_no_curvature)-f)^2);
append!(error_curvature, 0);

plot(error_deviation, error_curvature, label = "");
xlabel!("Sum-square deviation (y intercept = 0.3347)");
ylabel!("Sum-square curvature (x intercept = 1.9724e06)");
title!("Optimal tradeoff curve");
savefig("Optimal_tradeoff_curve.png")
u1 = 10e-7;
A_tilde = [1/sqrt(n)*I_n;sqrt(u1)*S];
y_tilde = [1/sqrt(n)*f;zeros(n-2,1)];
g1 = A_tilde\y_tilde;
u2 = 10e-5;
A_tilde = [1/sqrt(n)*I_n;sqrt(u2)*S];
y_tilde = [1/sqrt(n)*f;zeros(n-2,1)];
g2 = A_tilde\y_tilde;
u3 = 10e-4;
A_tilde = [1/sqrt(n)*I_n;sqrt(u3)*S];
y_tilde = [1/sqrt(n)*f;zeros(n-2,1)];
g3 = A_tilde\y_tilde;

scatter(f, label = "f", marker = (:star, 5), color = :black);
```

```
plot!(g_no_deviation, label = "mu = 0", line = (:solid, 2),);
plot!(g1, label = "mu = 10e-7", line = (:dash, 2),);
plot!(g2, label = "mu = 10e-5", line = (:dashdot, 2),);
plot!(g3, label = "mu = 10e-4", line = (:dashdotdot, 2),);
plot!(vec(g_no_curvature), label = "mu = inf", line = (:dot, 2),);
title!("Curves illustrating the trade-off");
savefig("Curves.png")
```

Note: Several exams had a typo that defined

$$c = \frac{1}{n-1} \sum_{i=2}^{n-1} \left(\frac{g_{i+1} - 2g_i + g_{i-1}}{1/n^2} \right)^2$$

instead of

$$c = \frac{1}{n-2} \sum_{i=2}^{n-1} \left(\frac{g_{i+1} - 2g_i + g_{i-1}}{1/n^2} \right)^2 .$$

The solutions above reflect the second definition. Full credit was given for answers consistent with either definition. *Some common errors*

- Several students tried to approximate f using low-degree polynomials. While fitting f to a polynomial does smooth f, it does not necessarily minimize $d + \mu c$ for some $\mu \geq 0$, nor does it illustrate the trade-off between curvature and deviation.
- In explaining how to find the g that minimizes $d + \mu c$ as $\mu \to \infty$, many people correctly observed that if $g \in \text{null}(S)$, then $c = 0$. For full credit, however, solutions had to show how to choose the vector in $\text{null}(S)$ that minimizes d.
- Many people chose to zoom in on a small section of the trade-off curve rather than plot the whole range from 0 to $\mu \to \infty$. Those solutions received full-credit provided they calculated the intersections with the axes (i.e. provided they found the minimum value for $d + \mu c$ when $\mu = 0$ and when $\mu \to \infty$).