Homework 1 solutions

3.270. Some properties of the product of two matrices. For each of the following statements, either show that it is true, or give a (specific) counterexample.

- If AB is full rank then A and B are full rank.
- If A and B are full rank then AB is full rank.
- If A and B have zero nullspace, then so does AB.
- If A and B are onto, then so is AB.

You can assume that $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$. Some of the false statements above become true under certain assumptions on the dimensions of A and B. As a trivial example, all of the statements above are true when A and B are scalars, *i.e.*, $n = m = p = 1$. For each of the statements above, find conditions on n, m, and p that make them true. Try to find the most general conditions you can. You can give your conditions as inequalities involving n, m, and p, or you can use more informal language such as "A and B are both skinny."

Solution. First note that an $m \times n$ matrix is full rank if and only if the maximum number of independent columns or rows is equal to $\min\{m, n\}$.

- If AB is full rank then A and B are full rank. *False.* Consider the following counter example:
$$A = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}, \quad AB = \begin{bmatrix} 1 & 1 \end{bmatrix}.$$
Clearly AB is full rank while B is not.
- If A and B are full rank then AB is full rank. *False.* Consider:
$$A = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad AB = 0.$$
Clearly, A and B are full rank while AB is not.
- If A and B have zero null space, then so does AB. *True.* The proof is easy. We will prove that $ABx = 0$ implies that $x = 0$ and hence $\text{null}(AB) = \{0\}$. If $ABx = 0$, since A has zero null space then $Bx = 0$. Now since B has zero null space this implies that $x = 0$ and we are done.
- If A and B are onto, then so is AB. *True.* We need to show that $y = ABx$ can be solved in x given any y. Suppose that $y \in \mathbb{R}^m$ is arbitrary. Since A is onto, then $y = A\tilde{x}$ holds for some $\tilde{x} \in \mathbb{R}^n$. Now consider the equation $\tilde{x} = Bx$. Since B is onto, then $\tilde{x} = Bx$ holds for some $x \in \mathbb{R}^p$. This proves that $y = ABx$ is solvable in x with $y = A\tilde{x}$ and $\tilde{x} = Bx$ and we are done.

Now we will find conditions under which the first two statements are correct. We will give these conditions based on the relative sizes of m, n and p, *i.e.*, when A is fat or skinny, B is fat or skinny, or AB is fat or skinny. We consider a square matrix to be both fat and skinny. There are 8 possible cases to check, but by using transposes we can reduce that down to 4 cases. For example lets consider the case when AB is full rank and fat, A is fat and B is fat we are considering wheiter A and B are full rank. Since AB is full rank, $(AB)^\mathsf{T}$ will also be full rank. We know that $(AB)^\mathsf{T} = B^\mathsf{T}A^\mathsf{T}$ so the same results apply for AB skinny, B and A skinny. First we consider the statement: "If AB is full rank then A and B are full rank."

- *A fat, B fat, AB fat* (or *A skinny, B skinny, AB skinny.*) The statement is not true for this case. Consider the counter example:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{\text{full rank}} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}}_{\text{not full rank}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{\text{full rank}}.$$

- *A fat, B skinny, AB fat* (or *A fat, B skinny, AB skinny.*) The statement is not true in this case. Consider:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}}_{\text{full rank}} \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}}_{\text{not full rank}} = \underbrace{\begin{bmatrix} 1 & 1 \end{bmatrix}}_{\text{full rank}}.$$

  However, if we add the constraint that AB is square then the statement becomes correct. To show this we use the facts that for a full rank fat matrix A all rows are independent, so $x^\mathsf{T}A = 0$ implies $x = 0$, and for a full rank skinny matrix B all columns are independent, so $Bx = 0$ implies that $x = 0$. We first prove by contradiction that AB full rank implies that A is full rank. If A (fat) is not full rank, then there exists an $x \neq 0$ such that $x^\mathsf{T}A = 0$, and therefore, $x^\mathsf{T}AB = 0$. This implies that the rows of AB (a square matrix) are dependent which is impossible since AB is full rank and we are done. Now we prove that B should be full rank as well. If B (skinny) is not full rank, then $Bx = 0$ for $x \neq 0$ which implies that $ABx = 0$, or the columns of AB (a full rank square matrix) are dependent which is a contradiction. Hence B is full rank too and we are done.

- *A skinny, B fat, AB fat* (or *A skinny, B fat, AB skinny.*) The statement is true in this case. First note that if AB is full rank then A should be square. We have

$$\mathbf{rank}(AB) \leq \mathbf{min}\{\mathbf{rank}(A), \mathbf{rank}(B)\}$$

  and since A is skinny and B is fat, $\mathbf{rank}(A) \leq n$ and $\mathbf{rank}(B) \leq n$ and therefore

$$\mathbf{rank}(AB) \leq n.$$

  Now since AB is full rank and fat, then $\mathbf{rank}(AB) = m$ so $m \leq n$. However, A is skinny so $m \geq n$ and therefore we can only have $m = n$ or that A is square. Now it is easy to prove that AB full rank implies that A and B are full rank. We first prove that A is full rank by contradiction. Suppose that A (square) is not full rank so there

exists a non-trivial linear combination of its rows that is equal to zero, *i.e.*, $x \neq 0$ and $x^\mathsf{T} A = 0$. Therefore, $x^\mathsf{T} AB = 0$ which implies that a linear combination of the rows of AB (a fat matrix) is zero which is impossible because AB is full rank. This shows that A should be full rank. Now we show that B should be full rank as well. Since A is full rank and square, then A^{-1} exists so $B = A^{-1}(AB)$. Suppose that B (fat) is not full rank so there exists an $x \neq 0$ such that $x^\mathsf{T} B = 0$ and therefore $x^\mathsf{T} A^{-1}(AB) = 0$. But $x^\mathsf{T} A^{-1}$ is nonzero because x is nonzero and A^{-1} is invertible, which implies that a linear combination of the rows of AB (a full rank fat matrix) is zero. This is impossible of course and we have shown by contradiction that B should be full rank and we are done.

- *A fat, B fat, AB skinny* (or *A skinny, B skinny, AB fat.*) If A is fat, B is fat and AB is skinny, then A, B and AB can only be square matrices. A being fat implies that $m \leq n$ and B being fat implies that $n \leq p$ and we get $p \geq m$. However, $p \leq m$ because AB is skinny, so we can only have $m = p$, and therefore $m = n$ as well. In other words, A, B and AB are square. As a result, this case (A square, B square, AB square) falls into the previous category (A skinny, B fat, AB fat) and hence the statement is true.

To summarize, the most general conditions for the statement to be true are:

- A fat, B skinny, AB square,
- A square, B fat, AB fat,
- A skinny, B square, AB skinny.

Comment: Another way to do this part:
The following inequalities are always true, regardless of the sizes of A, B and AB:

$$\mathbf{rank}(A) \leq \min\{m, n\}, \quad \mathbf{rank}(B) \leq \min\{n, p\}$$

$$\mathbf{rank}(AB) \leq \min\{\mathbf{rank}(A), \mathbf{rank}(B)\}$$

Since AB is full rank, we also have $\mathbf{rank}(AB) = \mathbf{min}\{m, p\}$. From this and the last inequality above we get the following:

$$\min\{m, p\} \leq \mathbf{rank}(A) \leq \min\{m, n\}, \quad \min\{m, p\} \leq \mathbf{rank}(B) \leq \min\{n, p\}$$

Now, with the three numbers m, n and p, there are six different cases. However, as mentioned before, we only need to check three cases, since the other three can be obtained by taking transposes. Using the above inequalities in each case, we get:

- $m \leq n \leq p$: $\mathbf{rank}(A) = m,\ m \leq \mathbf{rank}(B) \leq n$
  Thus in this case A will be full rank, but we can't say anything about B. The only way to be able to infer that B is also full rank is to have $m = n$. So the claim will be true if $m = n \leq p$.
- $m \leq p \leq n$: $\mathbf{rank}(A) = m,\ m \leq \mathbf{rank}(B) \leq p$
  Similar to the previous case, to be able to infer both A and B are full rank, we should have $m = p$. So the condition in this case will be $m = p \leq n$.

- $n \leq m \leq p$: $m \leq \mathbf{rank}(A) \leq n$, but $n \leq m$, so we must have $m = n \leq p$, yielding $\mathbf{rank}(A) = \mathbf{rank}(B) = m$.

Therefore, the most general conditions where the claim is true are:
$m = n \leq p,\ n = p \leq m,\ m = p \leq n$
Which are the same conditions as the ones obtained before.
Now we consider the second statement: "If A and B are full rank then AB is full rank." Again we consider different cases:

- *A fat, B fat, AB fat* (or *A skinny, B skinny, AB skinny.*) The statement is true in this case. Since AB is fat, we need to prove that $x^\mathsf{T}AB = 0$ implies that $x = 0$. But this is easy: $x^\mathsf{T}AB = 0$ implies that $x^\mathsf{T}A = 0$ (because B is fat and full rank) and $x^\mathsf{T}A = 0$ implies that $x = 0$ (because A is fat and full rank) and we are done.

- *A fat, B skinny, AB fat* (or *A fat, B skinny, AB skinny.*) The statement is not true in this case. Consider the counter example:

$$\underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{\text{full rank}} \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{\text{full rank}} = \underbrace{[0]}_{\text{not full rank}} .$$

- *A skinny, B fat, AB fat* (or *A skinny, B fat, AB skinny.*) The statement is not true in this case. Consider:

$$\underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\text{full rank}} \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{\text{full rank}} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}}_{\text{not full rank}} .$$

- *A fat, B fat, AB skinny* (or *A skinny, B skinny, AB fat.*) As shown previously, if A is fat, B is fat and AB is skinny, then A, B and AB can only be square matrices. Therefore, this case falls into the category of A fat, B fat, AB fat for which the statement is true.

To summarize, the statement is true only if

- A fat, B fat, AB fat,
- A skinny, B skinny, AB skinny.

4.580. Projection matrices. A matrix $P \in \mathbb{R}^{n\times n}$ is called a *projection matrix* if $P = P^\mathsf{T}$ and $P^2 = P$.

a) Show that if P is a projection matrix then so is $I - P$.

b) Suppose that the columns of $U \in \mathbb{R}^{n\times k}$ are orthonormal. Show that UU^T is a projection matrix. (Later we will show that the converse is true: every projection matrix can be expressed as UU^T for some U with orthonormal columns.)

c) Suppose $A \in \mathbb{R}^{n\times k}$ is full rank, with $k \leq n$. Show that $A(A^\mathsf{T}A)^{-1}A^\mathsf{T}$ is a projection matrix.

d) If $S \subseteq \mathbb{R}^n$ and $x \in \mathbb{R}^n$, the point y in S closest to x is called the *projection of x on S*. Show that if P is a projection matrix, then $y = Px$ is the projection of x on range(P). (Which is why such matrices are called projection matrices ...)

Solution.

a) To show that $I - P$ is a projection matrix we need to check two properties:

i. $I - P = (I - P)^\mathsf{T}$

ii. $(I - P)^2 = I - P$.

The first one is easy: $(I - P)^\mathsf{T} = I - P^\mathsf{T} = I - P$ because $P = P^\mathsf{T}$ (P is a projection matrix.) The show the second property we have

$$\begin{aligned}(I - P)^2 &= I - 2P + P^2 \\ &= I - 2P + P \qquad (\text{since } P = P^2) \\ &= I - P\end{aligned}$$

and we are done.

b) Since the columns of U are orthonormal we have $U^\mathsf{T}U = I$. Using this fact it is easy to prove that UU^T is a projection matrix, *i.e.*, $(UU^\mathsf{T})^\mathsf{T} = UU^\mathsf{T}$ and $(UU^\mathsf{T})^2 = UU^\mathsf{T}$. Clearly, $(UU^\mathsf{T})^\mathsf{T} = (U^\mathsf{T})^\mathsf{T}U^\mathsf{T} = UU^\mathsf{T}$ and

$$\begin{aligned}(UU^\mathsf{T})^2 &= (UU^\mathsf{T})(UU^\mathsf{T}) \\ &= U(U^\mathsf{T}U)U^\mathsf{T} \\ &= UU^\mathsf{T} \qquad (\text{since } U^\mathsf{T}U = I).\end{aligned}$$

c) First note that $\left(A(A^\mathsf{T}A)^{-1}A^\mathsf{T}\right)^\mathsf{T} = A(A^\mathsf{T}A)^{-1}A^\mathsf{T}$ because

$$\begin{aligned}\left(A(A^\mathsf{T}A)^{-1}A^\mathsf{T}\right)^\mathsf{T} &= (A^\mathsf{T})^\mathsf{T}\left((A^\mathsf{T}A)^{-1}\right)^\mathsf{T}A^\mathsf{T} \\ &= A\left((A^\mathsf{T}A)^\mathsf{T}\right)^{-1}A^\mathsf{T} \\ &= A(A^\mathsf{T}A)^{-1}A^\mathsf{T}.\end{aligned}$$

Also $\left(A(A^\mathsf{T}A)^{-1}A^\mathsf{T}\right)^2 = A(A^\mathsf{T}A)^{-1}A^\mathsf{T}$ because

$$\begin{aligned}\left(A(A^\mathsf{T}A)^{-1}A^\mathsf{T}\right)^2 &= \left(A(A^\mathsf{T}A)^{-1}A^\mathsf{T}\right)\left(A(A^\mathsf{T}A)^{-1}A^\mathsf{T}\right) \\ &= A\left((A^\mathsf{T}A)^{-1}A^\mathsf{T}A\right)(A^\mathsf{T}A)^{-1}A^\mathsf{T} \\ &= A(A^\mathsf{T}A)^{-1}A^\mathsf{T} \qquad (\text{since } (A^\mathsf{T}A)^{-1}A^\mathsf{T}A = I).\end{aligned}$$

d) To show that Px is the projection of x on range(P) we verify that the "error" $x - Px$ is orthogonal to *any* vector in range(P). Since range(P) is nothing but the span of the columns of P we only need to show that $x - Px$ is orthogonal to the columns of P, or in other words, $P^\mathsf{T}(x - Px) = 0$. But

$$\begin{aligned}P^\mathsf{T}(x - Px) &= P(x - Px) \qquad (\text{since } P = P^\mathsf{T}) \\ &= Px - P^2x \\ &= 0 \qquad (\text{since } P^2 = P)\end{aligned}$$

and we are done.

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 matlab, 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 matlab 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 in matlab 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.

4.670. 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 $x = A^{-1}y = R^{-1}(Q^{\mathsf{T}}y) = R^{-1}z$, where $z = Q^{\mathsf{T}}y$. In this exercise, you'll develop a method for computing $x = R^{-1}z$, *i.e.*, solving $Rx = z$, when R 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 *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.

Solution. Suppose that

$$z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}, \quad R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & r_{nn} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Consider the linear equation corresponding to the last (nth) row of R, *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, *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

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

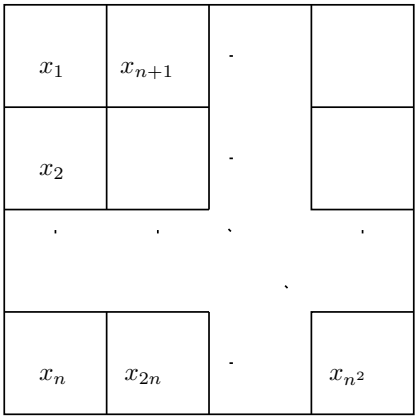
with x_n found from the previous step. In general, if $x_n, x_{n-1}, \ldots, x_{i+1}$ are known, x_i can be derived from the linear equation corresponding to the ith row of R as (assuming $r_{ii} \neq 0$)

$$x_i = \frac{1}{r_{ii}}\left(z_i - r_{i(i-1)}x_{i-1} - r_{i(i-2)i}x_{i-2} - \cdots - r_{in}x_n\right),$$

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 *back substitution*. This suggests the following simple algorithm:

$i := n$;
while $i \geq 1$
  if $r_{ii} \neq 0$
    $x_i := \frac{1}{r_{ii}}\left(z_i - \sum_{j=i+1}^{n} r_{ij}x_j\right)$;
  else
    unique solution does not exist; **break**;
  end
  $i := i - 1$;
end

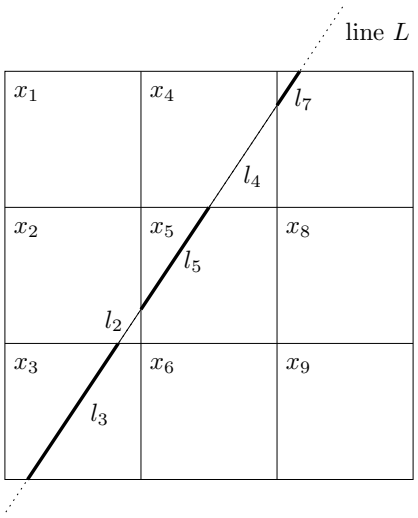
6.741. Image reconstruction from line integrals. In this problem we explore a simple version of a tomography problem. We consider a square region, which we divide into an $n \times n$ array of square pixels, as shown below.



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.m`, 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 the function `line_pixel_length.m` 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 matlab code:

```
tomodata

A=zeros(N,n_pixels^2);
for i=1:N
  L=line_pixel_length(lines_d(i),lines_theta(i),n_pixels);
  A(i,:)=L(:)';
end

x_ls=A\y;
X_ls=zeros(n_pixels,n_pixels);
X_ls(:)=x_ls;
```

```
figure(1)
colormap gray
imagesc(X_ls)
axis image
```

And here's the end result, the reconstructed image