Example 4.7.2: Let $A = \left[\begin{array}{c|c|c} 1 & 4 & 5 \\ 2 & 5 & 7 \\ 3 & 6 & 9 \end{array}\right]$. Since the sum of the first two columns equals the third column, $A * [1, 1, -1]$ is the zero vector. Thus $[1, 1, -1]$ is in Null A. By Equation 4.3, for any scalar α, $A * (\alpha [1, 1, -1])$ is also the zero vector, so $\alpha [1, 1, -1]$ is also in Null A. For example, $[2, 2, -2]$ is in Null A.

Problem 4.7.3: For each of the given matrices, find a nonzero vector in the null space of the matrix.

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

Here we make use of Equation 4.4:

Lemma 4.7.4: For any $R \times C$ matrix A and C-vector $\boldsymbol{v}$, a vector $\boldsymbol{z}$ is in the null space of A if and only if $A * (\boldsymbol{v} + \boldsymbol{z}) = A * \boldsymbol{v}$.

Proof

The statement is equivalent to the following statements:

1. if the vector $\boldsymbol{z}$ is in the null space of A then $A * (\boldsymbol{v} + \boldsymbol{z}) = A * \boldsymbol{v}$;
2. if $A * (\boldsymbol{v} + \boldsymbol{z}) = A * \boldsymbol{v}$ then $\boldsymbol{z}$ is in the null space of A.

For simplicity, we prove these two statements separately.

1. Suppose $\boldsymbol{z}$ is in the null space of A. Then

$$A * (\boldsymbol{v} + \boldsymbol{z}) = A * \boldsymbol{v} + A * \boldsymbol{z} = A * \boldsymbol{v} + \mathbf{0} = A * \boldsymbol{v}$$

2. Suppose $A * (\boldsymbol{v} + \boldsymbol{z}) = A * \boldsymbol{v}$. Then

$$\begin{aligned} A * (\boldsymbol{v} + \boldsymbol{z}) &= A * \boldsymbol{v} \\ A * \boldsymbol{v} + A * \boldsymbol{z} &= A * \boldsymbol{v} \\ A * \boldsymbol{z} &= \mathbf{0} \end{aligned}$$

□

4.7.2 The solution space of a matrix-vector equation

In Lemma 3.6.1 (Section 3.6.1), we saw that two solutions to a system of linear equations differ by a vector that solves the corresponding system of homogeneous equations. We restate and reprove the result in terms of matrix-vector equations:

Corollary 4.7.5: Suppose $\boldsymbol{u}_1$ is a solution to the matrix equation $A * \boldsymbol{x} = \boldsymbol{b}$. Then $\boldsymbol{u}_2$ is also a solution if and only if $\boldsymbol{u}_1 - \boldsymbol{u}_2$ belongs to the null space of A.

Proof

Since $A * \boldsymbol{u}_1 = \boldsymbol{b}$, we know that

$A * \boldsymbol{u}_2 = \boldsymbol{b}$ if and only if $A * \boldsymbol{u}_2 = A * \boldsymbol{u}_1$.

Applying Lemma 4.7.4 with $\boldsymbol{v} = \boldsymbol{u}_2$ and $\boldsymbol{z} = \boldsymbol{u}_1 - \boldsymbol{u}_2$, we infer:

$A * \boldsymbol{u}_2 = A * \boldsymbol{u}_1$ if and only if $\boldsymbol{u}_1 - \boldsymbol{u}_2$ is in the null space of A.

Combining these two statements proves the corollary. □

While studying a method for calculating the rate of power consumption for hardware components (Section 2.9.2), we asked about uniqueness of a solution to a system of linear equations. We saw in Corollary 3.6.4 that uniqueness depended on whether the corresponding homogeneous system have only the trivial solution. Here is the same corollary, stated in matrix terminology:

Corollary 4.7.6: Suppose a matrix-vector equation $A\boldsymbol{x} = \boldsymbol{b}$ has a solution. The solution is unique if and only if the null space of A consists solely of the zero vector.

Thus uniqueness of a solution comes down to the following question:

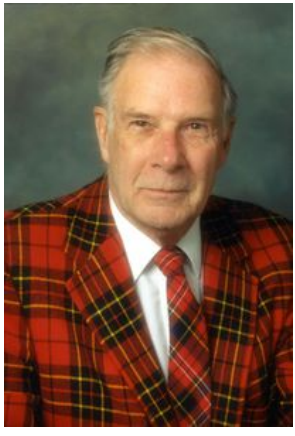
Question 4.7.7: How can we tell if the null space of a matrix consist solely of the zero vector?

This is just a restatement using matrix terminology, of Question 3.6.5, *How can we tell if a homogeneous linear system has only the trivial solution?*

While studying an attack on an authentication scheme in Section 2.9.7, we became interested in counting the solutions to a system of linear equations over $GF(2)$ (Question 2.9.18). In Section 3.6.1 we saw that this was equivalent to counting the solutions to a homogeneous system (Question 3.6.6). Here we restate this problem in terms of matrices:

Question 4.7.8: How can we find the cardinality of the null space of a matrix over $GF(2)$?

4.7.3 Introduction to error-correcting codes



Richard Hamming was getting, he later recalled, "very annoyed." He worked for Bell Laboratories in New Jersey but needed to use a computer located in New York. This was a very early computer, built using electromechanical relays, and it was somewhat unreliable. However, the computer could detect when an error occured, and when it did, it would restart the current computation. After three tries, however, it would go on to the next computation.

Hamming was, in his words, "low man on the totem pole", so he didn't get much use of the computer during the work week. However, nobody else was using it during the weekend. Hamming was allowed to submit a bunch of computations on Friday afternoon; the computer would run them during the weekend, and Hamming would be able to collect the results.

However, he came in one Monday to collect his results, and found that something went wrong, and all the computations failed. He tried again the following weekend—the same thing happened. Peeved, he asked himself: if the computer can detect that its input has an error, why can't it tell me where the error is?

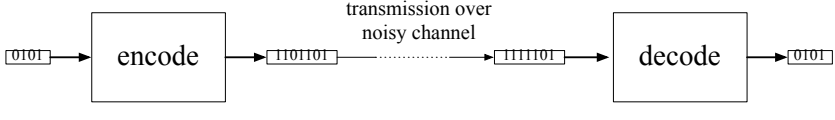
Hamming had long known one solution to this problem: replication. if you are worried about occasional bit errors, write your bit string three times: for each bit position, if the three bit strings differ in that position, choose the bit that occurs twice. However, this solution uses more bits than necessary.

As a result of this experience, Hamming invented *error-correcting codes*. The first code he invented is now called the *Hamming code* and is still used, e.g. in flash memory. He and other researchers subsequently discovered many other error-correcting codes. Error-correcting codes are ubiquitious today; they are used in many kinds of transmission (including WiFi, cell phones, communication with satellites and spacecraft, and digital television) and storage (RAM, disk drives, flash memory, CDs, and DVDs).

The Hamming code is what we now call a *linear binary block code*:

- *linear* because it is based on linear algebra,
- *binary* because the input and output are assumed to be in binary, and
- *block* because the code involves a fixed-length sequence of bits.

The transmission or storage of data is modeled by a *noisy channel*, a tube through which you can push vectors but which sometimes flips bits. A block of bits is represented by a vector over $GF(2)$. A binary block code defines a function $f : GF(2)^m \longrightarrow GF(2)^n$. (In the Hamming code, m is 4 and n is 7.)



When you have a block of m bits you want to be reliably received at the other end, you first use f to transform it to an n-vector, which you then push through the noisy channel. At the other end of the noisy channel, the recipient gets an n-vector that might differ from the original in some bit positions; the recipient must somehow figure out which bits were changed as the vector passed through the noisy channel.

We denote by $\mathcal{C}$ the set of encodings, the image of f—the set of n-vectors that can be injected into the noisy channel. The vectors of $\mathcal{C}$ are called *codewords*.

4.7.4 Linear codes

Let $\boldsymbol{c}$ denote the codeword injected into the noisy channel, and let $\tilde{\boldsymbol{c}}$ denote the vector (not necessarily a codeword) that comes out the other end. Ordinarily, $\tilde{\boldsymbol{c}}$ differs from $\boldsymbol{c}$ only in a small number of bit positions, the positions in which the noisy channel introduced errors. We write

$$\tilde{\boldsymbol{c}} = \boldsymbol{c} + \boldsymbol{e}$$

where $\boldsymbol{e}$ is the vector with 1's in the error positions. We refer to $\boldsymbol{e}$ as the *error vector*.

The recipient gets $\tilde{\boldsymbol{c}}$ and needs to figure out $\boldsymbol{e}$ in order to figure out $\boldsymbol{c}$. How?

In a linear code, the set $\mathcal{C}$ of codewords is the null space of a matrix H. This simplifies the job of the recipient. Using Equation 4.4, we see

$$H * \tilde{\boldsymbol{c}} = H * (\boldsymbol{c} + \boldsymbol{e}) = H * \boldsymbol{c} + H * \boldsymbol{e} = \boldsymbol{0} + H * \boldsymbol{e} = H * \boldsymbol{e}$$

because $\boldsymbol{c}$ is in the null space of H.

Thus the recipient knows something useful about $\boldsymbol{e}$: she knows $H * \boldsymbol{e}$ (because it is the same as $H * \tilde{\boldsymbol{c}}$, which she can compute). The vector $H * \boldsymbol{e}$ is called the *error syndrome*. If the error syndrome is the zero vector then the recipient assumes that $\boldsymbol{e}$ is all zeroes, i.e. that no error has been introduced. If the error syndrome is a nonzero vector then the recipient knows that an error has occured, i.e. that $\boldsymbol{e}$ is not all zeroes. The recipient needs to figure out $\boldsymbol{e}$ from the vector $H * \boldsymbol{e}$. The method for doing this depends on the particular code being used.

4.7.5 The Hamming Code

In the Hamming code, the codewords are 7-vectors, and

$$H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Notice anything special about the columns and their order?

Now suppose that the noisy channel introduces at most one bit error. Then $\boldsymbol{e}$ has only one 1. Can you determine the position of the bit error from the matrix-vector product $H * \boldsymbol{e}$?

Example 4.7.9: Suppose $\boldsymbol{e}$ has a 1 in its third position, $\boldsymbol{e} = [0, 0, 1, 0, 0, 0, 0]$. Then $H * \boldsymbol{e}$ is the third column of H, which is $[0, 1, 1]$.

As long as $\boldsymbol{e}$ has at most one bit error, the position of the bit can be determined from $H * \boldsymbol{e}$. This shows that the Hamming code allows the recipient to correct one-bit errors.

Quiz 4.7.10: Suppose $H * \boldsymbol{e}$ is $[1, 1, 0]$. What is $\boldsymbol{e}$?

Answer

$[0, 0, 0, 0, 0, 1, 0]$.

Quiz 4.7.11: Show that the Hamming code does not allow the recipient to correct two-bit errors: give two different error vectors, $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$, each with at most two 1's, such that $H * \boldsymbol{e}_1 = H * \boldsymbol{e}_2$.

Answer

There are many acceptable answers, e.g. $\boldsymbol{e}_1 = [1, 1, 0, 0, 0, 0, 0]$ and $\boldsymbol{e}_2 = [0, 0, 1, 0, 0, 0, 0]$ or $\boldsymbol{e}_1 = [0, 0, 1, 0, 0, 1, 0]$ and $\boldsymbol{e}_2 = [0, 1, 0, 0, 0, 0, 1]$.

Next we show that the Hamming code allows *detection* of errors as long as the number of errors is no more than two. Remember that the recipient assumes that no error has occured if $H * \boldsymbol{e}$ is the zero vector. Is there a way to set exactly two 1's in $\boldsymbol{e}$ so as to achieve $H * \boldsymbol{e} = \boldsymbol{0}$?

When $\boldsymbol{e}$ has two 1's, $H * \boldsymbol{e}$ is the sum of the two corresponding columns of H. If the sum of two columns is $\boldsymbol{0}$ then (by $GF(2)$ arithmetic) the two columns must be equal.

Example 4.7.12: Suppose $\boldsymbol{e} = [0, 0, 1, 0, 0, 0, 1]$. Then $H * \boldsymbol{e} = [0, 1, 1] + [1, 1, 1] = [1, 0, 0]$

Note, however, that a two-bit error can get misinterpreted as a one-bit error. In the example, if the recipient assumes at most one error, she will conclude that the error vector is $\boldsymbol{e} = [0, 0, 0, 1, 0, 0, 0]$.

In Lab 4.14, we will implement the Hamming code and try it out.

4.8 Computing sparse matrix-vector product

For computing products of matrices with vectors, we could use the linear-combinations or dot-products definitions but they are not very convenient for exploiting sparsity.

By combining the definition of dot-product with the dot-product definition of matrix-vector multiplication, we obtain the following equivalent definition.

Definition 4.8.1 (*Ordinary* Definition of Matrix-Vector Multiplication:): If M is an $R \times C$ matrix and $\boldsymbol{u}$ is a C-vector then $M * \boldsymbol{u}$ is the R-vector $\boldsymbol{v}$ such that, for each $r \in R$,

$$v[r] = \sum_{c \ C} M[r, c]\boldsymbol{u}[c] \tag{4.5}$$

The most straightforward way to implement matrix-vector multiplication based on this definition is:

1 for each i in R:
2 $\quad v[i] := \sum_{j \ C} M[i,j]u[j]$

However, this doesn't take advantage of the fact that many entries of M are zero and do not even appear in our sparse representation of M. We could try implementing the sum in Line 2 in a clever way, omitting those terms corresponding to entries of M that do not appear in our sparse representation. However, our representation does not support doing this efficiently. The more general idea is sound, however: iterate over the entries of M that are actually represented.

The trick is to initialize the output vector $\boldsymbol{v}$ to the zero vector, and then iterate over the nonzero entries of M, adding terms as specified by Equation 4.5.

1 initialize $\boldsymbol{v}$ to zero vector
2 for each pair (i,j) such that the sparse representation specifies $M[i,j]$,
3 $\quad v[i] = v[i] + M[i,j]u[j]$

A similar algorithm can be used to compute a vector-matrix product.

Remark 4.8.2: This algorithm makes no effort to exploit sparsity in the vector. When doing matrix-vector or vector-matrix multiplication, it is not generally worthwhile to try to exploit sparsity in the vector.

Remark 4.8.3: There could be zeroes in the output vector but such zeroes are considered "accidental" and are so rare that it is not worth trying to notice their occurence.

4.9 The matrix meets the function

4.9.1 From matrix to function

For every matrix M, we can use matrix-vector multiplication to define a function $\boldsymbol{x} \mapsto M * \boldsymbol{x}$. The study of the matrix M is in part the study of this function, and vice versa. It is convenient to have a name by which we can refer to this function. There is no traditional name for this function; just in this section, we will refer to it by f_M. Formally, we define it as follows: if M is an $R \times C$ matrix over a field $\mathbb{F}$ then the function $f_M : \mathbb{F}^C \longrightarrow \mathbb{F}^R$ is defined by $f_M(\boldsymbol{x}) = M * \boldsymbol{x}$

This is not a traditional definition in linear algebra; I introduce it here for pedagogical purposes.

Example 4.9.1: Let M be the matrix

	#	@	?
a	1	2	3
b	10	20	30

Then the domain of the function f_M is $\mathbb{R}^{\{\#,@,?\}}$ and the co-domain is $\mathbb{R}^{\{a,b\}}$. The image, for

example, of the vector $\begin{array}{ccc}\texttt{\#} & \texttt{@} & \texttt{?} \\ \hline 2 & 2 & \text{-}2\end{array}$ is the vector $\begin{array}{cc}\texttt{a} & \texttt{b} \\ \hline 0 & 0\end{array}$

Problem 4.9.2: Recall that M^T is the transpose of M. The function corresponding to M^T is f_{M^T}

1. What is the domain of f_{M^T}?
2. What is the co-domain?
3. Give a vector in the domain of f_{M^T} whose image is the all-zeroes vector.

4.9.2 From function to matrix

Suppose we have a function $f_M : \mathbb{F}^A \longrightarrow \mathbb{F}^B$ corresponding to some matrix M but we don't happen to know the matrix M. We want to compute the matrix M such that $f_M(\boldsymbol{x}) = M * \boldsymbol{x}$.

Let's first figure out the column-label set for M. Since the domain of f_M is $\mathbb{F}^A$, we know that $\boldsymbol{x}$ is an A-vector. For the product $M * \boldsymbol{x}$ to even be legal, we need the column-label set of M to be A.

Since the co-domain of f_M is $\mathbb{F}^B$, we know that the result of multiplying M by $\boldsymbol{x}$ must be a B-vector. In order for that to be the case, we need the row-label set of M to be B.

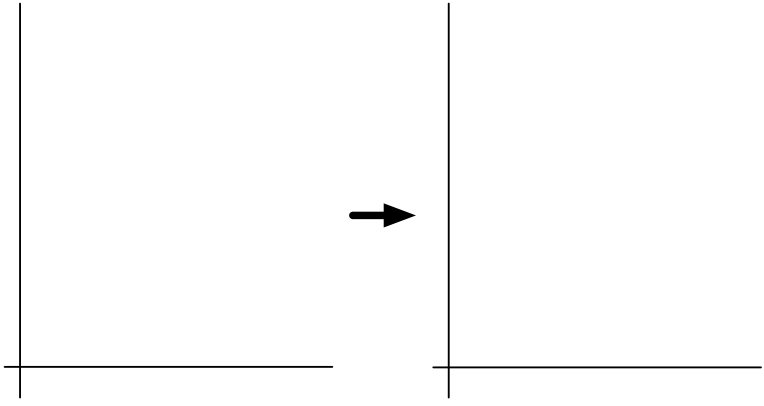
So far, so good. We know M must be a $B \times A$ matrix. But what should its entries be? To find them, we use the linear-combinations definition of matrix-vector product.

Remember the standard generators for $\mathbb{F}^A$: for each element $a \in A$, there is a generator $\boldsymbol{e}_a$ that maps a to one and maps every other element of A to zero. By the linear-combinations definition, $M * \boldsymbol{e}_a$ is column a of M. This shows that column a of M must equal $f_M(\boldsymbol{e}_a)$.

4.9.3 Examples of deriving the matrix

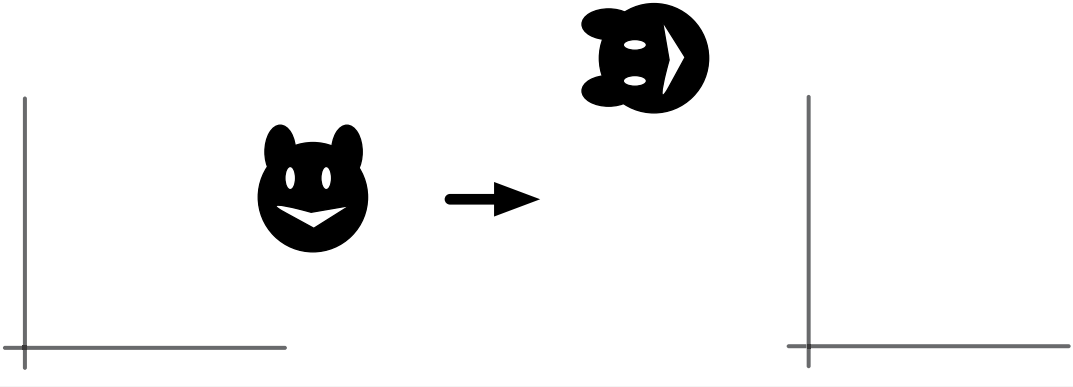
In this section, we give some examples illustrating how one derives the matrix from a function, *assuming* that there is some matrix M such that the function is $\boldsymbol{x} \mapsto M * \boldsymbol{x}$. *Warning:* In at least one of these examples, that assumption is not true.

Example 4.9.3: Let $s(\cdot)$ be the function from $\mathbb{R}^2$ to $\mathbb{R}^2$ that scales the x-coordinate by 2.



Assume that $s([x,y]) = M * [x,y]$ for some matrix M. The image of $[1,0]$ is $[2,0]$ and the image of $[0,1]$ is $[0,1]$, so $M = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$.

Example 4.9.4: Let $r_{90}(\cdot)$ be the function from $\mathbb{R}^2$ to $\mathbb{R}^2$ that rotates points in 2D by ninety degrees counterclockwise around the origin.



Let's assume for now that $r_{90}([x,y]) = M * [x,y]$ for some matrix M. To find M, we find the image under this function of the two standard generators $[1,0]$ and $[0,1]$.

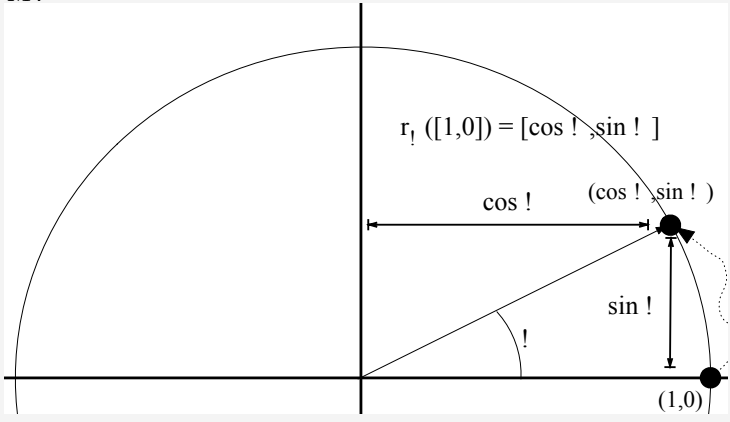
Rotating the point $[1,0]$ by ninety degrees about the origin yields $[0,1]$, so this must be the first column of M.

Rotating the point $[0,1]$ by ninety degrees yields $[-1,0]$, so this must be the second column of M. Therefore $M = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$.

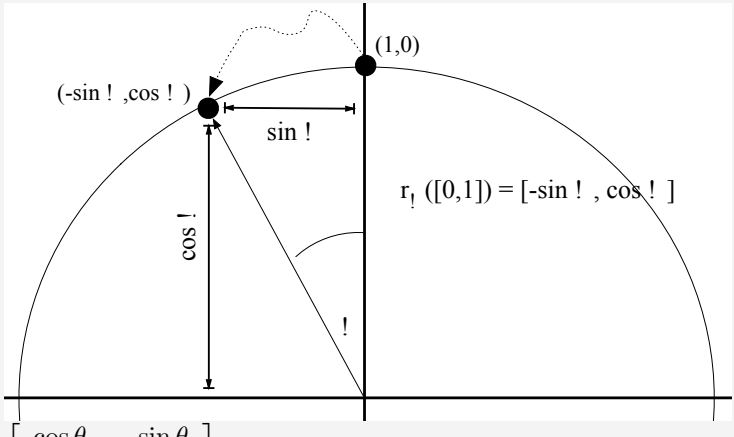
Example 4.9.5: For an angle θ, let $r_\theta(\cdot)$ be the function from $\mathbb{R}^2$ to $\mathbb{R}^2$ that rotates points

around the origin counterclockwise by θ. Assume $r_\theta([x,y]) = M * [x,y]$ for some matrix M.

Rotating the point $[1,0]$ by θ gives us the point $[\cos\theta, \sin\theta]$, which must therefore be the first column of M.

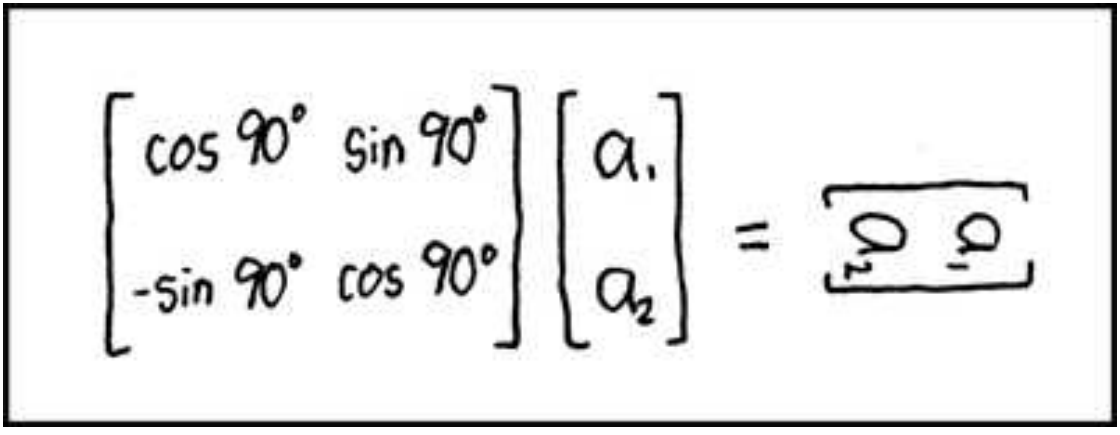


Rotating the point $[0,1]$ by θ gives us the point $[-\sin\theta, \cos\theta]$, so this must be the second column of M.



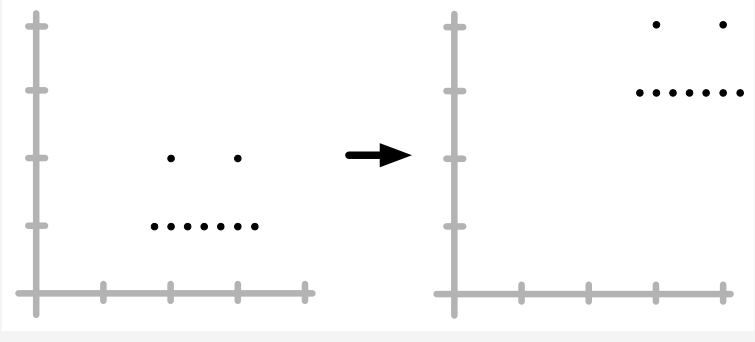
Thus $M = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$.

For example, for rotating by thirty degrees, the matrix is $\begin{matrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{matrix}$. Finally, we have caught up with complex numbers, for which rotation by a given angle is simply multiplication (Section 1.4.10).



Matrix Transform (`http://xkcd.com/824`)

Example 4.9.6: Let $t(\cdot)$ be the function from $\mathbb{R}^2$ to $\mathbb{R}^2$ that translates a point one unit to the right and two units up.



Assume that $t([x, y]) = M * [x, y]$ for some matrix M. The image of $[1, 0]$ is $[2, 2]$ and the image of $[0, 1]$ is $[1, 3]$, so $M = \begin{bmatrix} 2 & 1 \\ 2 & 3 \end{bmatrix}$.

4.10 Linear functions

In each of the examples, we *assumed* that the function could be expressed in terms of matrix-vector multiplication, but this assumption turns out not to be valid in all these examples. How can we tell whether a function can be so expressed?

4.10.1 Which functions can be expressed as a matrix-vector product

In Section 3.4, we identified three properties, Property V1, Property V2, and Property V3, that hold of

- the span of some vectors, and

- the solution set of a homogeneous linear system.

We called any set of vectors satisfying Properties V1, V2, and V3 a *vector space*.

Here we take a similar approach. In Section 4.6.5, we proved two algebraic properties of matrix-vector multiplication. We now use those algebraic properties to define a special kind of function, *linear functions*.

4.10.2 Definition and simple examples

Definition 4.10.1: Let $\mathcal{U}$ and $\mathcal{V}$ be vector spaces over a field $\mathbb{F}$. A function $f : \mathcal{U} \longrightarrow \mathcal{V}$ is called a *linear* function if it satisfies the following two properties:

Property L1: For any vector $\boldsymbol{u}$ in the domain of f and any scalar α in $\mathbb{F}$,

$$f(\alpha\,\boldsymbol{u}) = \alpha\, f(\boldsymbol{u})$$

Property L2: For any two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ in the domain of f,

$$f(\boldsymbol{u} + \boldsymbol{v}) = f(\boldsymbol{u}) + f(\boldsymbol{v})$$

(A synonym for *linear function* is *linear transformation*.)
Let M be an $R \times C$ matrix over a field $\mathbb{F}$, and define

$$f : \mathbb{F}^C \longrightarrow \mathbb{F}^R$$

by $f(\boldsymbol{x}) = M * \boldsymbol{x}$. The domain and co-domain are vector spaces. By Proposition 4.6.13, the function f satisfies Properties L1 and L2. Thus f is a linear function. We have proved:

Proposition 4.10.2: For any matrix M, the function $\boldsymbol{x} \mapsto M * \boldsymbol{x}$ is a linear function.

Here is a special case.

Lemma 4.10.3: For any C-vector $\boldsymbol{a}$ over $\mathbb{F}$, the function $f : \mathbb{F}^C \longrightarrow \mathbb{F}$ defined by $f(\boldsymbol{x}) = \boldsymbol{a} \cdot \boldsymbol{x}$ is a linear function.

Proof

Let A be the $\{0\} \times C$ matrix whose only row is $\boldsymbol{a}$. Then $f(\boldsymbol{x}) = A * \boldsymbol{x}$, so the lemma follows from Proposition 4.10.2. □

Bilinearity of dot-product Lemma 4.10.3 states that, for any vector $\boldsymbol{w}$, the function $\boldsymbol{x} \mapsto \boldsymbol{w} \cdot \boldsymbol{x}$ is a linear function of $\boldsymbol{x}$. Thus the dot-product function $f(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} \cdot \boldsymbol{y}$ is linear in its first argument (i.e. if we plug in a vector for the second argument). By the symmetry of the

dot-product (Proposition 2.9.21), the dot-product function is also linear in its second argument. We say that the dot-product function is *bilinear* to mean that it is linear in each of its arguments.

Example 4.10.4: Let $\mathbb{F}$ be any field. The function from $\mathbb{F}^2$ to $\mathbb{F}$ defined by $(x, y) \mapsto x + y$ is a linear function. You can prove this using bilinearity of dot-product.

Quiz 4.10.5: Show that the function with domain $\mathbb{R}^2$ defined by $[x, y] \mapsto xy$ is *not* a linear function by giving inputs for which the function violates either Property L1 or Property L2.

Answer

$$f([1,1] + [1,1]) = f([2,2]) = 4$$
$$f([1,1]) + f([1,1]) = 1 + 1$$

Quiz 4.10.6: Show that rotation by ninety degrees, $r_{90}(\cdot)$, is a linear function.

Answer

The scalar-multiplication property, Property L1, is proved as follows:

$$\begin{aligned} \alpha\, f([x,y]) &= \alpha\, [-y, x] \\ &= [-\alpha y, \alpha x\,] \\ &= f([\alpha x, \alpha y\,]) \\ &= f(\alpha\, [x,y]) \end{aligned}$$

The vector-addition property, Property L2, is proved similiarly:

$$\begin{aligned} f([x_1,y_1]) + f([x_2,y_2]) &= [-y_1, x_1] + [-y_2, x_2] \\ &= [-(y_1+y_2), x_1 + x_2] \\ &= f([x_1+x_2, y_1+y_2]) \end{aligned}$$

Exercise 4.10.7: Define $g : \mathbb{R}^2 \longrightarrow \mathbb{R}^3$ by $g([x,y]) = [x,y,1]$. Is g a linear function? If so, prove it. If not, give a counterexample.

Exercise 4.10.8: Define $h : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$ to be the function that takes a point $[x, y]$ to its reflection about the y-axis. Give an explicit (i.e. algebraic) definition of h. Is it a linear function? Explain your answer.

Problem 4.10.9: In at least one of the examples in Section 4.9.3, the function cannot be written as $f(\boldsymbol{x}) = M * \boldsymbol{x}$. Which one? Demonstrate using a numerical example that the function does not satisfy the Properties L1 and L2 that define linear functions.

4.10.3 Linear functions and zero vectors

Lemma 4.10.10: If $f : \mathcal{U} \longrightarrow \mathcal{V}$ is a linear function then f maps the zero vector of $\mathcal{U}$ to the zero vector of $\mathcal{V}$.

Proof

Let $\mathbf{0}$ denote the zero vector of $\mathcal{U}$, and let $\mathbf{0}_{\mathcal{V}}$ denote the zero vector of $\mathcal{V}$.

$$f(\mathbf{0}) = f(\mathbf{0} + \mathbf{0}) = f(\mathbf{0}) + f(\mathbf{0})$$

Subtracting $f(\mathbf{0})$ from both sides, we obtain

$$\mathbf{0}_{\mathcal{V}} = f(\mathbf{0})$$

□

Definition 4.10.11: Analogous to the null space of a matrix (Definition 4.7.1), we define the *kernel* of a linear function f to be $\{\boldsymbol{v} \ : \ f(\boldsymbol{v}) = \mathbf{0}\}$. We denote the kernel of f by Ker f.

Lemma 4.10.12: The kernel of a linear function is a vector space.

Problem 4.10.13: Prove Lemma 4.10.12 by showing that Ker f satisfies Properties V1, V2, and V3 of vector spaces (Section 3.4).

4.10.4 What do linear functions have to do with lines?

Suppose $f : \mathcal{U} \longrightarrow \mathcal{V}$ is a linear function. Let $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ be two vectors in $\mathcal{U}$, and consider a linear combination $\alpha_1 \boldsymbol{u}_1 + \alpha_2 \boldsymbol{u}_2$ and its image under f.

$$\begin{aligned} f(\alpha_1 \boldsymbol{v}_1 + \alpha_2 \boldsymbol{v}_2) &= f(\alpha_1 \boldsymbol{v}_1) + f(\alpha_2 \boldsymbol{v}_2) && \text{by Property L2} \\ &= \alpha_1 f(\boldsymbol{v}_1) + \alpha_2 f(\boldsymbol{v}_2) && \text{by Property L1} \end{aligned}$$

We interpret this as follows: the image of a linear combination of $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ is the corresponding linear combination of $f(\boldsymbol{u}_1)$ and $f(\boldsymbol{u}_2)$.

What are the geometric implications?

Let's focus on the case where the domain $\mathcal{U}$ is $\mathbb{R}^n$. The line through the points $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ is the affine hull of $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, i.e. the set of all affine combinations:

$$\{\alpha_1 \boldsymbol{u}_1 + \alpha_2 \boldsymbol{u}_2 \ : \ \alpha_1, \alpha_2 \in \mathbb{R}, \alpha_1 + \alpha_2 = 1\}$$

What is the set of images under f of all these affine combinations? It is

$$\{f(\alpha_1 \boldsymbol{u}_1 + \alpha_2 \boldsymbol{u}_2) \ : \ \alpha_1, \alpha_2 \in \mathbb{R}, \alpha_1 + \alpha_2 = 1\}$$

which is equal to

$$\{\alpha_1 f(\boldsymbol{u}_1) + \alpha_2 f(\boldsymbol{u}_2) \ : \ \alpha_1, \alpha_2 \in \mathbb{R}, \alpha_1 + \alpha_2 = 1\}$$

which is the set of all affine combinations of $f(\boldsymbol{u}_1)$ and $f(\boldsymbol{u}_2)$.

This shows:

The image under f of the line through $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ is the "line" through $f(\boldsymbol{u}_1)$ and $f(\boldsymbol{u}_2)$.

The reason for the scare-quotes is that f might map $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ to the same point! The set of affine combinations of two identical points is the set consisting just of that one point.

The argument we have given about the image of a linear combination can of course be extended to handle a linear combination of more than two vectors.

Proposition 4.10.14: For a linear function f, for any vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ in the domain of f and any scalars $\alpha_1, \ldots, \alpha_n$,

$$f(\alpha_1 \boldsymbol{u}_1 + \cdots + \alpha_n \boldsymbol{u}_n) = \alpha_1 f(\boldsymbol{u}_1) + \cdots + \alpha_n f(\boldsymbol{u}_n)$$

Therefore the image under a linear function of any flat is another flat.

4.10.5 Linear functions that are one-to-one

Using the notion of kernel, we can give a nice criterion for whether a linear function is one-to-one.

Lemma 4.10.15 (One-to-One Lemma): A linear function is one-to-one if and only if its kernel is a trivial vector space.

Proof

Let $f : \mathcal{V} \longrightarrow \mathcal{W}$ be a linear function. We prove two directions.

Suppose Ker f contains some nonzero vector $\boldsymbol{v}$, so $f(\boldsymbol{v}) = \mathbf{0}_{\mathcal{V}}$. By Lemma 4.10.10, $f(\mathbf{0}) = \mathbf{0}_{\mathcal{V}}$ as well, so f is not one-to-one.

Suppose Ker $f = \{\mathbf{0}\}$. Let $\boldsymbol{v}_1, \boldsymbol{v}_2$ be any vectors such that $f(\boldsymbol{v}_1) = f(\boldsymbol{v}_2)$. Then $f(\boldsymbol{v}_1) - f(\boldsymbol{v}_2) = \mathbf{0}_{\mathcal{V}}$ so, by linearity, $f(\boldsymbol{v}_1 - \boldsymbol{v}_2) = \mathbf{0}_{\mathcal{V}}$, so $\boldsymbol{v}_1 - \boldsymbol{v}_2 \in$ Ker f. Since Ker f consists solely of $\mathbf{0}$, it follows that $\boldsymbol{v}_1 - \boldsymbol{v}_2 = \mathbf{0}$, so $\boldsymbol{v}_1 = \boldsymbol{v}_2$. □

This simple lemma gives us a fresh perspective on the question of uniqueness of solution to a linear system. Consider the function $f(\boldsymbol{x}) = A * \boldsymbol{x}$. Solving a linear system $A * \boldsymbol{x} = \boldsymbol{b}$ can be interpreted as finding a pre-image of $\boldsymbol{b}$ under f. If a pre-image exists, it is guaranteed to be unique if f is one-to-one.

4.10.6 Linear functions that are onto?

The One-to-One Lemma gives us a nice criterion for determining whether a linear function is one-to-one. What about onto?

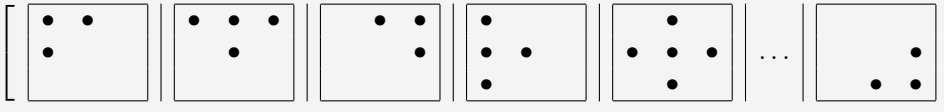
Recall that the *image* of a function f with domain $\mathcal{V}$ is the set $\{f(v) \ : \ v \in \mathcal{V}\}$. Recall that a function f being onto means that the image of the function equals the co-domain.

Question 4.10.16: How can we tell if a linear function is onto?

When $f : \mathcal{V} \longrightarrow \mathcal{W}$ is a linear function, we denote the image of f by Im f. Thus asking whether f is onto is asking whether Im $f = \mathcal{W}$.

Example 4.10.17: (Solvability of *Lights Out*) Can 3×3 *Lights Out* be solved from any initial configuration? (Question 2.8.5).

As we saw in Example 4.5.5 (Page 195), we can use a matrix to address *Lights Out*. We construct a matrix M whose columns are the button vectors:

$M =$ 

The set of solvable initial configurations (those from which it is possible to turn out all lights) is the set of all linear combinations of these columns, the column space of the matrix. We saw in Example 3.2.14 (Page 154) that, in the case of 2×2 *Lights Out*, every initial configuration is solvable in this case. What about 3×3 *Lights Out*?

Let $D = \{(0,0), \ldots, (2,2)\}$. Let $f : GF(2)^D \longrightarrow GF(2)^D$ be defined by $f(\boldsymbol{x}) = M * \boldsymbol{x}$. The set of solvable initial configurations is the image of f. The set of all initial configurations is the co-domain of f. Therefore, the question of whether every position is solvable is exactly the question of whether f is onto.

We can make one step towards answering Question 4.10.16.

Lemma 4.10.18: The image of a linear function is a subspace of the function's co-domain.

Proof

Let $f : \mathcal{V} \longrightarrow \mathcal{W}$ be a linear function. Clearly Im f is a subset of $\mathcal{W}$. To show that Im f is a subspace of $\mathcal{W}$, we must show that Im f satisfies Properties V1, V2, and V3 of a vector space.

- *V1:* We saw in Lemma 4.10.10 that f maps the zero vector of $\mathcal{V}$ to the zero vector of $\mathcal{W}$, so the zero vector of $\mathcal{W}$ belongs to Im f.
- *V2:* Let $\boldsymbol{w}$ be a vector in Im f. By definition of Im f, there must be a vector $\boldsymbol{v}$ in $\mathcal{V}$ such that $f(\boldsymbol{v}) = \boldsymbol{w}$. For any scalar α,

  $$\alpha\,\boldsymbol{w} = \alpha\, f(\boldsymbol{v}) = f(\alpha\,\boldsymbol{v})$$

  so $\alpha\,\boldsymbol{w}$ is in Im f.
- *V3:* Let $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ be vectors in Im f. By definition of Im f, there must be vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ in $\mathcal{V}$ such that $f(\boldsymbol{v}_1) = \boldsymbol{w}_1$ and $f(\boldsymbol{v}_2) = \boldsymbol{w}_2$. By Property L1 of linear functions, $\boldsymbol{w}_1 + \boldsymbol{w}_2 = f(\boldsymbol{v}_1) + f(\boldsymbol{v}_2) = f(\boldsymbol{v}_1 + \boldsymbol{v}_2)$, so $\boldsymbol{w}_1 + \boldsymbol{w}_2$ is in Im f.

□

The complete answer to Question 4.10.16 must wait until Chapter 6.

4.10.7 A linear function from $\mathbb{F}^C$ to $\mathbb{F}^R$ can be represented by a matrix

Suppose $f : \mathbb{F}^C \longrightarrow \mathbb{F}^R$ is a linear function. We can use the method of Section 4.9.2 to obtain a matrix M: for each $c \in C$, column c of M is the image under f of the standard generator $\boldsymbol{e}_c$.

How do we know the resulting matrix M satisfies $f(\boldsymbol{x}) = M * \boldsymbol{x}$? Linearity! For any vector $\boldsymbol{x} \in \mathbb{F}^C$, for each $c \in C$, let α_c be the value of entry c of $\boldsymbol{x}$. Then $\boldsymbol{x} = \sum_{c \ C} \alpha_c\, \boldsymbol{e}_c$. Because f is linear, $f(\boldsymbol{x}) = \sum_{c \ C} \alpha_c f(\boldsymbol{e}_c)$.

On the other hand, by the linear-combinations definition of matrix-vector multiplication, $M * \boldsymbol{x}$ is the linear combination of M's columns where the coefficients are the scalars α_c (for $c \in C$). We defined M to be the matrix whose column c is $f(\boldsymbol{e}_c)$ for each $c \in C$, so $M * \boldsymbol{x}$ also equals $\sum_{c \ C} \alpha_c f(\boldsymbol{e}_c)$. This shows that $f(\boldsymbol{x}) = M * \boldsymbol{x}$ for every vector $\boldsymbol{x} \in \mathbb{F}^C$.

We summarize this result in a lemma.

Lemma 4.10.19: If $f : \mathbb{F}^C \longrightarrow \mathbb{F}^R$ is a linear function then there is an $R \times C$ matrix M over $\mathbb{F}$ such that $f(\boldsymbol{x}) = M * \boldsymbol{x}$ for every vector $\boldsymbol{x} \in \mathbb{F}^C$.

4.10.8 Diagonal matrices

Let $d_1, \ldots, d_n$ be real numbers. Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ be the function such that $f([x_1, \ldots, x_n]) = [d_1 x_1, \ldots, d_n x_n]$. The matrix corresponding to this function is

$$\begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix}$$

Such a matrix is called a *diagonal* matrix because the only entries allowed to be nonzero form a diagonal.

Definition 4.10.20: For a domain D, a $D \times D$ matrix M is a *diagonal* matrix if $M[r, c] = 0$ for every pair $r, c \in D$ such that $r = c$.

Diagonal matrices are very important in Chapters 11 and 12.

Quiz 4.10.21: Write a procedure `diag(D, entries)` with the following spec:

- *input:* a set `D` and a dictionary `entries` mapping `D` to elements of a field
- *output:* a diagonal matrix such that entry `(d,d)` is `entries[d]`

Answer

```
def diag(D, entries):
 return Mat((D,D), {(d,d):entries[d] for d in D})
```

A particularly simple and useful diagonal matrix is the *identity* matrix, defined in Section 4.1.5. For example, here is the $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\} \times \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ identity matrix:

```
      a b c
     -------
 a   |  1 0 0
 b   |  0 1 0
 c   |  0 0 1
```

Recall that we refer to it as $\mathbb{1}_D$ or just $\mathbb{1}$.

Why is it called the identity matrix? Consider the function $f : \mathbb{F}^D \longrightarrow \mathbb{F}^D$ defined by $f(\boldsymbol{x}) = \mathbb{1} * \boldsymbol{x}$. Since $\mathbb{1} * \boldsymbol{x} = \boldsymbol{x}$, the function f is the identity function on $\mathbb{F}^D$.

4.11 Matrix-matrix multiplication

We can also multiply a pair of matrices. Suppose A is an $R \times S$ matrix and B is an $S \times T$ matrix. Then it is legal to multiply A times B, and the result is a $R \times T$ matrix. The traditional way of writing "A times B" is simply AB, with no operator in between the matrices.