Chapter 10

Transformations in two dimensions

Goal: Discuss how we use linear transformations in 2D in computer graphics, how homogeneous coordinates let us describe translations as well.

10.1 Introduction

When we think about taking some object for which we have a geometric model (i.e., for which we know the positions of a collection of points) and 'putting it in a scene.' we typically need to do three things: *move* the object to some location, *scale* it up or down so that it fits well with the other objects in the scene, and *rotate* it until it has the right orientation. These operations — translation, rotation, and scaling — are part of every graphics system. Both rotation and scaling are *linear transformations* on the coordinates of the object's points. Recall that a linear transformation

$$T : \mathbf{R^2} \to \mathbf{R^2}$$

is one for which $T(\mathbf{v} + \alpha\mathbf{w}) = T(\mathbf{v}) + \alpha T(\mathbf{w})$ for any two vectors $\mathbf{v}$ and $\mathbf{w}$ in $\mathbf{R^2}$, and any real number α.

The definition of linearity tells us that the zero vector $\mathbf{0}$ is always sent to the zero vector: if we choose $\mathbf{v} = \mathbf{w} = \mathbf{0}$ and $\alpha = 1$, the definition tells us that

$$T(\mathbf{0}) = T(\mathbf{0} + 1\mathbf{0}) = T(\mathbf{0}) + 1T(\mathbf{0}) = T(\mathbf{0}) + T(\mathbf{0}).$$

Subtracting $T(\mathbf{0})$ from the first and last parts of this chain gives us $\mathbf{0} = T(\mathbf{0})$.

Inline Exercise 10.1: Suppose T is linear. Insert $\alpha = 1$ in the definition of linearity; what does it say? Insert $\mathbf{v} = \mathbf{0}$ in the definition. What does it say?

This means that *translation* — moving every point of the plane by the same amount — is in general *not* a linear transformation (except in the special case of translation-by-zero, i.e., leaving all points where they are). Shortly we'll describe a trick for putting the Euclidean plane into $\mathbf{R^3}$ (but *not* as the $z = 0$ plane as is usually done); once we do this, we'll see that certain linear transformations on $\mathbf{R^3}$ end up performing translations on this embedded plane.

But for now, let's discuss linear transformations in the plane. We will assume that you have *some* familiarity with linear transformations already; indeed, the serious student of computer graphics should, at some point, study linear algebra carefully. But one can learn a great deal of graphics with only a modest amount of the subject, which we'll summarize here briefly.

In the first few sections, we'll use the convention of most linear-algebra texts: the vectors are arrows at the origin, and we'll think of the vector $\begin{bmatrix} u \\ v \end{bmatrix}$ as being identified with the point (u, v). Later we'll return to the point-vector distinction.

In the course of discussing these transformations, we'll make two substantial digressions: the first to discuss the *singular value decomposition* of a matrix, which arises naturally in the study of matrix transformations, and the second to discuss the set $\mathbf{SO}(2)$ of all rotations of the plane and the exponential map, in preparation for a corresponding discussion for transformations in three dimensions.

10.1.1 Five examples

We begin with five examples of linear transformations in the plane; we'll refer to these by the names $T_1, \ldots, T_5$ throughout the chapter.

Example 1: Rotation. Let

$$T_1 : \mathbf{R^2} \to \mathbf{R^2} : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} \cos 30^\circ & -\sin 30^\circ \\ \sin 30^\circ & \cos 30^\circ \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

Recall that $\mathbf{e}_1$ denotes the vector $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$, and $\mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$; this transformation sends $\mathbf{e}_1$ to the vector $\begin{bmatrix} \cos 30^\circ \\ \sin 30^\circ \end{bmatrix}$, and $\mathbf{e}_2$ to $\begin{bmatrix} -\sin 30^\circ \\ \cos 30^\circ \end{bmatrix}$, which are vectors that are thirty degrees counterclockwise from the x and y axes, respectively, as shown in Figure 1.1.

There's nothing special about the number thirty in this example; by replacing thirty degrees by any angle, you can build a transformation that

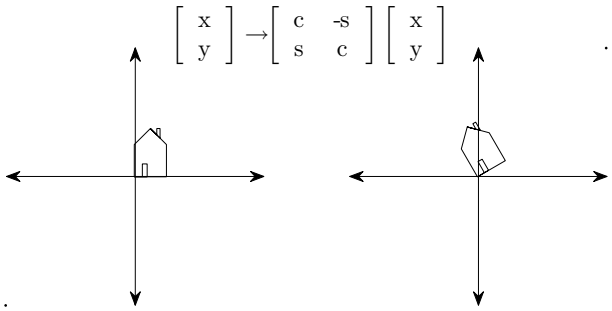

Figure 10.1: The transformation T_1 rotates the entire plane by 30 degrees counterclockwise; we show a house-shaped polygon in the domain, and its position in the codomain to help you see the transformation. The letters c and s denote the cosine and sine of thirty degrees, respectively.

fig:rot30

rotates things counterclockwise by that angle.

Inline Exercise 10.2: Write down the transformation that rotates everything in the plane by 180 degrees counterclockwise. Actually compute the sines and cosines so that you end up with a matrix filled with numbers in your answer. Apply this transformation to the corners of the unit square, $(0,0), (1,0), (0,1)$, and $(1,1)$.

Example 2: Nonuniform scaling. Let

$$T_2 : \mathbf{R^2} \to \mathbf{R^2} : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3x \\ 2y \end{bmatrix}.$$

This transformation stretches everything by a factor of three in the x-direction, and a factor of two in the y-direction, as shown in Figure 1.2. If the stretch amounts along the two axes were both three, we'd say that the transformation "scaled things up by three" and is a *uniform scaling transformation*; T_2 represents a generalization of this idea: rather than scaling uniformly in each direction, it's called a *nonuniform scaling transformation*, or, less formally, a *nonuniform scale*.

One again the example generalizes: by placing numbers other than 2 and 3 along the diagonal of the matrix, we can scale each axis by any amount we please; these scaling amounts can include zero and negative numbers.

Inline Exercise 10.3: Write down the matrix for a uniform scale by -1; how does your answer relate to the answer to the last exercise? Can you explain?

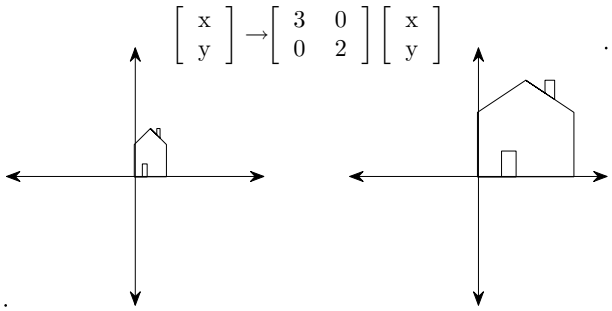

Figure 10.2: T_2 stretches things along the x-axis by three, and along the y-axis by two. This is called a nonuniform scaling transformation.

fig:nonu-scale

Inline Exercise 10.4: Write down a transformation matrix that scales in x by zero, and in y by one. Informally describe what the associated transformation does to the house.

Example 3: Shearing. Let

$$T_3 : \mathbf{R^2} \to \mathbf{R^2} : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x+2y \\ y \end{bmatrix}.$$

As Figure 1.3 shows, T_3 preserves "height" along the y-axis, but moves points parallel to the x-axis, with the amount of movement determined by the y-value. The x-axis itself remains fixed. Such a transformation is called a *shearing transformation*.

Inline Exercise 10.5: Generalize to build a transformation that keeps the y-axis fixed, but shears vertically instead of horizontally.

Example 4: A general transformation. Let

$$T_4 : \mathbf{R^2} \to \mathbf{R^2} : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 1 & -1 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Figure 1.4 shows the effects of T_4. It distorts the house figure, but not by just a rotation or scaling or shearing along the coordinate axes.

Example 5: A degenerate (or *singular* transformation Let

$$T_5 : \mathbf{R^2} \to \mathbf{R^2} : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 1 & -1 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x-y \\ 2x-2y \end{bmatrix}.$$

Figure 1.5 shows why we call this transformation *degenerate*: unlike the others, it collapses the whole two-dimensional plane down to a one-dimensional

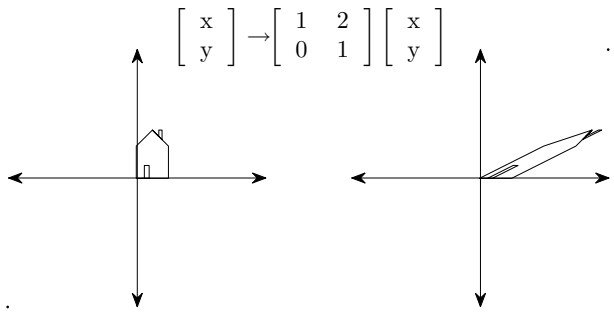

Figure 10.3: A "shearing" transformation, in which the x-axis remains fixed, but points above it are sheared to the right, while points below are sheared to the left; note that heights of points do not change — only their horizontal positions.

fig:shear

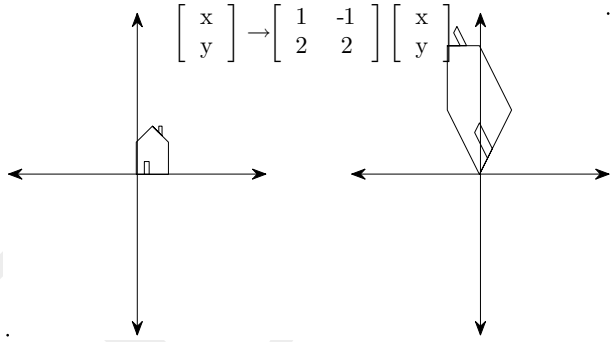

Figure 10.4: A "general" transformation, represented by a matrix that has none of the special forms of the previous examples. Looking at the house before and after transformation, we see that it's been quite distorted, in a way that's hard to describe simply, as we've done for the earlier examples.

fig:general-transform

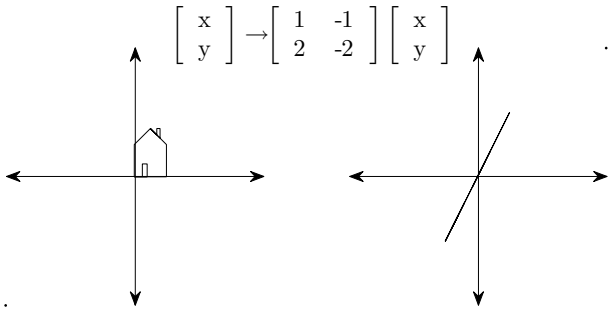

Figure 10.5: A degenerate transformation: the entire plane is collapsed into a single line, consisting of points where y is twice x. The "house" figure is collapsed to a jumble of overlapping segments.

fig:degenerate

subspace, i.e., a line. There's no longer a nice correspondence between points in the domain and points in the codomain: certain points in the codomain don't correspond to *any* point in the domain any more; others correspond to many points in the domain. Such a transformation is also called *singular*, as is the matrix defining it. Those familiar with linear algebra will note that this is equivalent to saying that the determinant of $\begin{bmatrix}1 & -1\\2 & -2\end{bmatrix}$ is zero, or saying that its columns are linearly dependent.

10.1.2 Important facts about transformations

We'll describe here several properties of linear transformations from $\mathbf{R^2}$ to $\mathbf{R^2}$; these properties are important in part because they all generalize: they apply (in some form) to transformations from $\mathbf{R}^n$ to $\mathbf{R}^k$ for any n and k. We'll mostly be concerned with values of n and k between one and four; for this section, we'll concentrate on $n = k = 2$.

Multiplication by a matrix is a linear transformation

If M is a 2×2 matrix, then the function T_{M} defined by

$$T_{\mathrm{M}} : \mathbf{R^2} \to \mathbf{R^2} : \mathbf{x} \mapsto \mathbf{Mx}$$

is linear. All five examples above demonstrate this.

For nondegenerate transformations, lines are sent to lines, as T_1 through T_4 show; for degenerate ones, a line may be sent to a single point; for instance, T_5 sends the line consisting of all vectors of the form $\begin{bmatrix}b\\b\end{bmatrix}$ to the zero vector.

Because multiplication by a matrix M is always a linear transformation, we'll call T_{M} the *transformation associated to the matrix* M.

Multiplication by a matrix is the *only* linear transformation

In $\mathbf{R}^n$, it turns out that for *every* linear transform T, there's a matrix M with $T(\mathbf{x}) = \mathrm{M}\mathbf{x}$. We'll see shortly how to find M, given T, even if T is expressed in some other way.

Because of this, we can all M the *matrix associated to the transformation* T. In fact, the result is stronger: there's exactly one matrix associated to any linear transformation. Put differently, if someone tells you that

$$T(\mathbf{x}) = \mathbf{M}\mathbf{x}$$

and also that

$$T(\mathbf{x}) = \mathbf{S}\mathbf{x}$$

then you can conclude that $\mathbf{M} = \mathbf{S}$.

The matrix $\mathbf{I}$, with ones on the diagonal and zeroes off the diagonal, is called the *identity matrix*; the associated transformation

$$T(\mathbf{x}) = \mathbf{I}\mathbf{x}$$

is special: it's the transformation that leaves every vector x unchanged.

Inline Exercise 10.6: There is an identity matrix of every size: a 1×1 identity, a 2×2 identity, and so on. Write out the first three.

Function composition and matrix multiplication are related

If M and $\mathbf{K}$ are 2×2 matrices, then they define transformations T_{M} and $T_{\mathbf{K}}$. When we compose these, we get the transformation

$$T_{\mathrm{M}} \circ T_{\mathbf{K}} : \mathbf{R^2} \to \mathbf{R^2} : \mathbf{x} \mapsto T_{\mathrm{M}}(T_{\mathbf{K}}(\mathbf{x})) = T_{\mathrm{M}}(\mathbf{K}\mathbf{x}) = \mathbf{M}(\mathbf{K}\mathbf{x}) = (\mathbf{M}\mathbf{K})\mathbf{x} = T_{\mathbf{M}\mathbf{K}}(\mathbf{x}). \tag{10.1}$$

In other words, the composed transformation is also a matrix transformation, with matrix $\mathbf{M}\mathbf{K}$. Note that when we write $T_{\mathrm{M}}(T_{\mathbf{K}}(\mathbf{x}))$, the transformation $T_{\mathbf{K}}$ is applied *first*. So, for example, if we look at the transformation $T_2 \circ T_3$, it first shears the house and *then* scales the result nonuniformly.

Inline Exercise 10.7: Describe the appearance of the house after transforming it by $T_1 \circ T_2$ and after transforming it by $T_2 \circ T_1$.

Matrix inverse and inverse functions are related

A matrix M is *invertible* if there's a matrix B with the property that $\mathrm{BM} = \mathrm{MB} = \mathrm{M}$. If such a matrix exists, it's denoted M^{-1}.

If M is invertible, and $S(\mathbf{x}) = \mathrm{M}^{-1}\mathbf{x}$, then S is the inverse function of T_{M}, i.e.,

$$S(T(x)) = x \quad \text{and} \tag{10.2}$$

$$T(S(x)) = x. \tag{10.3}$$

Inline Exercise 10.8: Using Equation 1.1, explain why Equation holds.

If M not invertible, then T_{M} has no inverse.

Let's look at our examples. The matrix for T_1 has an inverse: simply replace 30 by -30 in all the entries; the resulting transformation rotates clockwise by 30 degrees; performing one rotation and then the other effectively does nothing (i.e., is the identity transformation). The inverse for the matrix for T_2 is diagonal, with entries $1/3$ and $1/2$. The inverse of the matrix for T_3 is $\begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix}$ (note the minus sign). The associated transformation also shears parallel to the x-axis, but vectors in the upper half-plane are moved to the *left*, which undoes the moving-to-the-right done by T_3.

For these first three, it was fairly easy to guess the inverse matrices, because we could understand how to invert the transformation. The inverse of the matrix for T_4 is

$$\frac{1}{4}\begin{bmatrix} 2 & 1 \\ -2 & 1 \end{bmatrix}$$

which we computed using a general rule for inverses of 2×2 matrices (the only such rule worth memorizing):

$$\begin{bmatrix} a & c \\ b & d \end{bmatrix}^{-1} = \frac{1}{ad - bc}\begin{bmatrix} d & -c \\ -b & a \end{bmatrix}. \tag{10.4}$$

Finally, for T_5, the matrix has no inverse; if it did, the function T_5 would be invertible: it would be possible to identify, for each point in the codomain, a single point in the domain that's sent there. But we've already seen this isn't possible.

Inline Exercise 10.9: Apply the formula from equation 1.4 to the matrix for T_5 to attempt to compute its inverse. What goes wrong? Your answer will give you a general test for noninvertibility of a 2×2 matrix.

Exercise 10.1: Multiply $M = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$ by the expression given in Equation 1.4 for its inverse to verify that the product really is the identity.

Finding the matrix for a transformation

We've said that every linear transformation really is just multiplication by some matrix, but how do you *find* that matrix? Suppose, for instance, that we'd like to find a linear transformations to flip our house across the y-axis, so that it ends up on the left side of the y-axis. (You may be able to guess the transformation that does this in your head, and the associated matrix, but we'll work through it directly.)

The key idea is this: if you know where the transformation sends $\mathbf{e}_1$ and $\mathbf{e}_2$, you know the matrix. Why? We know that the transformation must have the form

$$T\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}a & b\\c & d\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}; \tag{10.5}$$

we just don't know the values of $a, b, c,$ and d. Well, $T(\mathbf{e}_1)$ is then

$$T\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}a & b\\c & d\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}a\\c\end{bmatrix}. \tag{10.6}$$

Similarly, $T(\mathbf{e}_2)$ is the vector $\begin{bmatrix}b\\d\end{bmatrix}$. So knowing $T(\mathbf{e}_1)$ and $T(\mathbf{e}_2)$ tells us all the matrix entries. Applying this to the problem of flipping over the house, we know that $T(\mathbf{e}_1) = -\mathbf{e}_1$, because we want a point on the positive x-axis to be sent to the corresponding point on the negative x-axis, so $a = -1$ and $c = 0$. On the other hand, $T(\mathbf{e}_2) = \mathbf{e}_2$, because every vector on the y-axis should be left untouched, so $b = 0$ and $d = 1$. Thus the matrix for the "house flip" transformation is just

$$\begin{bmatrix}-1 & 0\\0 & 1\end{bmatrix}.$$

Inline Exercise 10.10: (a) Use this idea to find a matrix transformation sending $\mathbf{e}_1$ to $\begin{bmatrix}0\\4\end{bmatrix}$ and $\mathbf{e}_2$ to $\begin{bmatrix}1\\1\end{bmatrix}$; verify that it works. (b) Use the relationship of matrix inverse to the inverse of a transform, and the formula for the inverse of a 2×2 matrix, to find a transformation sending $\begin{bmatrix}0\\4\end{bmatrix}$ to $\mathbf{e}_1$ and $\begin{bmatrix}1\\1\end{bmatrix}$ to $\mathbf{e}_2$ to; again, verify that it works.

As the previous inline exercise shows, we now have the tools to send the *standard basis vectors* $\mathbf{e}_1$ and $\mathbf{e}_2$ to any two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, and vice versa (provided that $\mathbf{v}_1$ and $\mathbf{v}_2$ are independent, i.e., neither is a multiple of the other). We can combine this with the idea that composition of linear transformations (performing one after the other) corresponds to multiplication of matrices and thus create a solution to a rather general problem.

Problem: Given independent vectors $\mathbf{u}_1$ and $\mathbf{u}_2$, and any two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, find a linear transformation, in matrix form, that sends $\mathbf{u}_1$ to $\mathbf{v}_1$ and $\mathbf{u}_2$ to $\mathbf{v}_2$.

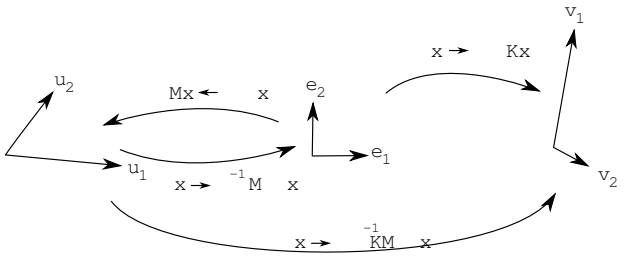

Figure 10.6: Multiplication by the matrix $\mathbf{M}$ takes $\mathbf{e}_1$ and $\mathbf{e}_2$ to $\mathbf{u}_1$ and $\mathbf{u}_2$, respectively, so multiplying $\mathbf{M}^{-1}$ does the opposite. Multiplying by $\mathbf{K}$ takes $\mathbf{e}_1$ and $\mathbf{e}_2$ to $\mathbf{v}_1$ and $\mathbf{v}_2$, so multiplying first by $\mathbf{M}^{-1}$ and then by $\mathbf{K}$, i.e., multiplying by $\mathbf{K}\mathbf{M}^{-1}$ takes $\mathbf{u}_1$ to $\mathbf{e}_1$ to $\mathbf{v}_1$, and similarly for $\mathbf{u}_2$.

fig:two-step-xform

Solution: Let $\mathbf{M}$ be the matrix whose columns are $\mathbf{u}_1$ and $\mathbf{u}_2$. Then

$$T : \mathbf{R^2} \to \mathbf{R^2} : \mathbf{x} \mapsto \mathbf{M}\mathbf{x}$$

sends $\mathbf{e}_1$ to $\mathbf{u}_1$ and $\mathbf{e}_2$ to $\mathbf{u}_2$ (see Figure 1.6). Therefore

$$S : \mathbf{R^2} \to \mathbf{R^2} : \mathbf{x} \mapsto \mathbf{M}^{-1}\mathbf{x}$$

sends $\mathbf{u}_1$ to $\mathbf{e}_1$ and $\mathbf{u}_2$ to $\mathbf{e}_2$.

Now let $\mathbf{K}$ be the matrix with columns $\mathbf{v}_1$ and $\mathbf{v}_2$. The transformation

$$R : \mathbf{R^2} \to \mathbf{R^2} : \mathbf{x} \mapsto \mathbf{K}\mathbf{x}$$

sends $\mathbf{e}_1$ to $\mathbf{v}_1$ and $\mathbf{e}_2$ to $\mathbf{v}_2$.

If we apply first S and then R to $\mathbf{u}_1$, it will be sent to $\mathbf{e}_1$ (by S), and thence to $\mathbf{v}_1$ by R; a similar argument applies to $\mathbf{u}_2$. But let's write this out:

$$\begin{aligned} R(S(\mathbf{x})) &= R(\mathbf{M}^{-1}\mathbf{x}) && (10.7)\\ &= \mathbf{K}(\mathbf{M}^{-1}\mathbf{x}) && (10.8)\\ &= (\mathbf{K}\mathbf{M}^{-1})\mathbf{x} && (10.9) \end{aligned}$$

Thus the matrix for the transformation sending the $\mathbf{u}$s to the $\mathbf{v}$s is just $\mathbf{K}\mathbf{M}^{-1}$.

Let's make this concrete with an example: we'll find a matrix sending

$$\mathbf{u}_1 = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \qquad \text{and } \mathbf{u}_2 \qquad = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

to

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \text{and } \mathbf{v}_2 \qquad = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

respectively. Following the pattern above, the matrices $\mathbf{M}$ and $\mathbf{K}$ are

$$\mathbf{M} = \begin{bmatrix} 2 & 1 \\ 3 & -1 \end{bmatrix}$$
$$\mathbf{K} = \begin{bmatrix} 1 & 2 \\ 1 & -1 \end{bmatrix}.$$

Using the matrix inversion formula (equation 1.4), we find

$$\mathbf{M}^{-1} = \frac{-1}{5}\begin{bmatrix} -1 & -1 \\ -3 & 2 \end{bmatrix}$$

so that the matrix for the overall transformation is

$$\begin{aligned} \mathbf{J} = \mathbf{K}\mathbf{M}^{-1} &= \begin{bmatrix} 1 & 2 \\ 1 & -1 \end{bmatrix} \cdot \frac{-1}{5}\begin{bmatrix} -1 & -1 \\ -3 & 2 \end{bmatrix} \\ &= \begin{bmatrix} 7/5 & -3/5 \\ -2/5 & 3/5 \end{bmatrix}. \end{aligned}$$

Inline Exercise 10.11: Verify that the transformation associated to the matrix $\mathbf{J}$ really does send $\mathbf{u}_1$ to $\mathbf{v}_1$, and $\mathbf{u}_2$ to $\mathbf{v}_2$.

Inline Exercise 10.12: Let $\mathbf{u}_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ and $\mathbf{u}_2 = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$; pick any two nonzero vectors you like as $\mathbf{v}_1$ and $\mathbf{v}_2$, and find the matrix transformation that sends each $\mathbf{u}_i$ to the corresponding $\mathbf{v}_i$.

The recipe for building matrix transformations above shows the following: every linear transformation from $\mathbf{R^2}$ to $\mathbf{R^2}$ is determined by its values on 2 independent vectors. In fact, this is a far more general property: any linear transformation from $\mathbf{R^2}$ to $\mathbf{R}^k$ is determined by its values on two independent vectors, and indeed, any linear transformation from $\mathbf{R}^n$ to $\mathbf{R}^k$ is determined by its values on n independent vectors (where to make sense of these, we need to extend our definition of "independence" to more than two vectors, which we'll do presently).

10.1.3 A different kind of transformation

We tend to think about linear transformations as moving points around (but leaving the origin fixed); we'll do that often. But equally important is their use in changing coordinate systems. If we have two coordinate systems on $\mathbf{R^2}$ with the same origin (we'll address this restriction later) as in Figure 1.7 then every arrow has coordinates in the red system and in the blue one. The two red coordinates can be written as a vector, as can the two blue coordinates. The vector $\mathbf{u}$, for instance, has coordinates $\begin{bmatrix} 3 \\ 2 \end{bmatrix}$ in the red system, and $\begin{bmatrix} -0.2 \\ 3.6 \end{bmatrix}$ in the blue system.

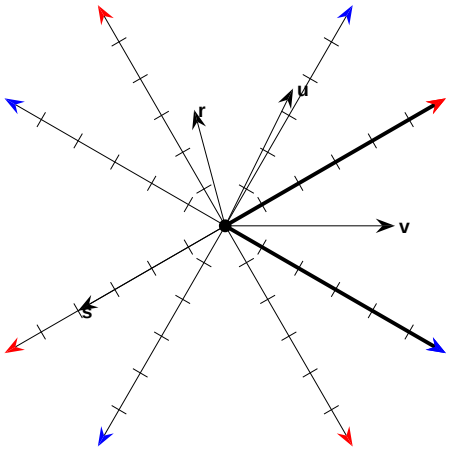

Figure 10.7: Two different coordinate systems for $\mathbf{R^2}$; the vector $\mathbf{u}$, expressed in the red coordinate system, has coordinates 3 and 2, while in the blue coordinate system, its coordinates are approximately -0.2 and 3.6, where we've drawn, in each case, the positive side of the first coordinate axis in bold. Both the red and the blue coordinates of $\mathbf{u}$ are different from its coordinates in the standard xy-coordinate system, which we have not drawn). You should determine the approximate red and blue coordinates of the vectors $\mathbf{t}$, $\mathbf{s}$, and $\mathbf{r}$ as well.

fig:two-coords

Inline Exercise 10.13: Use a ruler to find the coords of t, r and s in each of the two coordinate systems.

If we tabulated every imaginable arrow's coordinates in the red and blue system, we could use this tabulation to convert from red to blue coordinates. But it's far simpler than that: the conversion from red coordinates to blue coordinates is *linear*, and can be expressed by a matrix transformation. In this example, the matrix is

$$\mathbf{M} = \frac{1}{2}\begin{bmatrix} 1 & -\sqrt{3} \\ \sqrt{3} & 1 \end{bmatrix}.$$

Multiplying M by the coordinates of $\mathbf{u}$ in the red system gets us

$$\begin{aligned} \mathbf{v} &= \mathbf{M}\mathbf{u} \\ &= \frac{1}{2}\begin{bmatrix} 1 & -\sqrt{3} \\ \sqrt{3} & 1 \end{bmatrix}\begin{bmatrix} 3 \\ 2 \end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix} 3-2\sqrt{3} \\ 3\sqrt{3}+2 \end{bmatrix} \qquad\qquad \approx \begin{bmatrix} -0.2 \\ 3.6 \end{bmatrix} \end{aligned}$$

which is the coordinate vector for u in the blue system.

Inline Exercise 10.14: Confirm, for each of the other arrows in the drawing, that the same transformation converts red to blue coordinates.

We'll return to this idea later, but for now, there's one more important notion: in the special case where we want to go from the usual coordinates on a vector to its coordinates in some coordinate system with basis vectors $\mathbf{u}_1, \mathbf{u}_2$, which are *unit vectors* and *mutually perpendicular*, the transformation matrix is one whose *rows* are the transposes of $\mathbf{u}_1$ and $\mathbf{u}_2$.

For example, if $\mathbf{u}_1 = \begin{bmatrix} 3/5 \\ 4/5 \end{bmatrix}$ and $\mathbf{u}_2 = \begin{bmatrix} -4/5 \\ 3/5 \end{bmatrix}$ (check for yourself that these are unit-length and perpendicular), then the vector $\mathbf{v} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$, expressed in $\mathbf{u}$-coordinates, is

$$\begin{bmatrix} 3/5 & 4/5 \\ -4/5 & 3/5 \end{bmatrix}\begin{bmatrix} 4 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ -2 \end{bmatrix}.$$

Verify for yourself that these really *are* the $\mathbf{u}$-coordinates of $\mathbf{v}$, i.e., that the vector $\mathbf{v}$ really is the same as $4\mathbf{u}_1 + (-2)\mathbf{u}_2$.

10.1.4 Important properties of matrices

Because matrices are so closely tied to linear transformations, and because linear transformations are so important in graphics, we should note a few important properties of matrices.

The first is that *diagonal* matrices — ones with zeroes everywhere except on the diagonal, like the one we saw in the scaling transformation T_2 at the start of the chapter — correspond to remarkably simple transformations: they simply scale up or down each axis by some amount (although if the amount is a negative number, the corresponding axis is also flipped). Because of this simplicity, we'll try to understand other transformations in terms of these diagonal matrices.

Second, if the columns of the matrix $\mathbf{M}$ are $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k \in R^n$, and they are pairwise orthogonal unit vectors, then $\mathbf{M}^t\mathbf{M} = \mathbf{I}_k$, the $k \times k$ identity matrix.

Inline Exercise 10.15: What will MM^t look like? What size will it be? Where will its nonzero entries be? Experiment with a 3×2 matrix $\mathbf{M}$ whose columns are $\mathbf{e}_1, \mathbf{e}_3 \in \mathbf{R^3}$ to find out.

In the special case where $k = n$, such a matrix is called *orthogonal*. If the vectors form a positively-oriented coordinate system, then the matrix is said to be a *special orthogonal* matrix. In $\mathbf{R^2}$, such a matrix must be a rotation matrix like the one in T_1; in $\mathbf{R^3}$, the transformation associated to such a matrix corresponds to rotation around some vector by some amount[1].

Less familiar to most students, but of enormous importance in much graphics research, is the *singular value decomposition* of a matrix, which says, informally, that if we have a transformation T represented by a matrix M, then if we're willing to use a different coordinate system on the domain, and a different coordinate system on the codomain, then the transformation simply looks like a nonuniform (or possibly uniform) scaling transformation.

The theorem says this:

Every $n \times k$ matrix $\mathbf{M}$ can be factored in the form

$$\mathbf{M} = \mathbf{UDV}^t,$$

where $\mathbf{U}$ is $n \times r$ (where $r = \min(n, k)$) with orthonormal columns, $\mathbf{D}$ is $r \times r$ diagonal (i.e., only entries of the form d_{ii} can be nonzero), and $\mathbf{V}$ is $r \times k$ with orthonormal columns (see Figure 1.8

By convention, the entries of $\mathbf{D}$ are required to be in non-increasing order (i.e., $d_{1,1} \geq d_{2,2} \geq d_{3,3} \ldots$), and are indicated by single subscripts (i.e., we write d_1 instead of $d_{1,1}$). They are called the *singular values* of $\mathbf{M}$. It turns out that M is degenerate (i.e., singular) exactly if any singular value is zero. As a general guideline, if ratio of the largest to the smallest singular values is very large, then numerical computations with the matrix are likely to be unstable.

[1]Rotation about a vector in $\mathbf{R^3}$ is better expressed as rotation *in a plane*, so instead of speaking about rotation about z, we speak of rotation in the xy-plane. We can then say that a special orthogonal matrix in $\mathbf{R}^4$ corresponds to a sequence of two rotations in two planes in 4-space.

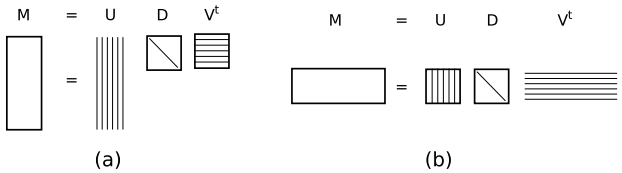

Figure 10.8: (a) An $n \times k$ matrix, with $n > k$, factors as a product of an $n \times n$ matrix with orthonormal columns (indicated by the vertical stripes on the first rectangle), a diagonal $k \times k$ matrix, and a $k \times k$ matrix with orthonormal rows (indicated by the horizontal stripes), which we write as $\mathbf{UDV}^t$, where $\$Umat$ and $\mathbf{V}$ have orthonormal *columns*. (b) An $n \times k$ matrix with $n < k$ is written as a similar product; note that the diagonal matrix, in both cases, is square, and its size is the smaller of n and k.

fig:svd-shape

Inline Exercise 10.16: The singular value decomposition is not unique; explain why if we negate the first row of $\mathbf{V}^t$ and the first column of $\mathbf{U}$ in the SVD of a matrix $\mathbf{M}$, the result is still an SVD for $\mathbf{M}$.

In the special case where $n = k$ (the one we most often encounter), the matrices $\mathbf{U}$ and $\mathbf{V}$ are both square, and represent change-of-coordinate transformations in the domain and codomain. Thus we can see the transformation

$$T(\mathbf{x}) = \mathbf{Mx}$$

as a sequence of three steps: multiplication by $\mathbf{V}^t$ converts $\mathbf{x}$ to $\mathbf{v}$-coordinates; multiplication by $\mathbf{D}$ amounts to a possibly non-uniform scaling along each axis; multiplication by $\mathbf{U}$ treats the resulting entries as coordinates in the $\mathbf{u}$-coordinate system, which then are transformed back to standard coordinates.

The singular value decomposition and outer products

Consider the SVD of an $n \times n$ square matrix

$$\mathbf{M} = \mathbf{UDV}^t.$$

We'll let $\mathbf{u}_1, \ldots, \mathbf{u}_n$ be the columns of $\mathbf{U}$, and $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be the columns of $\mathbf{V}$. We'll see that the matrix $\mathbf{M}$ can be written as a sum of matrices of the form $\mathbf{u}_i\mathbf{v}_i^t$, which is called the *outer product* of $\mathbf{u}_i$ and $\mathbf{v}_i$.

Inline Exercise 10.17: What are the dimensions (i.e., the number of rows and columns) of the matrix $\mathbf{u}_i$? What about $\mathbf{v}_i^t$? What are the dimensions of $\mathbf{u}_i\mathbf{v}_i^t$?

A single example tells the whole story; we've built one in which the numbers work out nicely:

$$\frac{1}{5\sqrt{2}}\begin{bmatrix}1 & -17\\ 18 & -6\end{bmatrix} = \mathbf{UDV}^t = \begin{bmatrix}3/5 & -4/5\\ 4/5 & 3/5\end{bmatrix}\begin{bmatrix}3 & 0\\ 0 & 2\end{bmatrix}\begin{bmatrix}1/\sqrt{2} & 1/\sqrt{2}\\ -1/\sqrt{2} & 1/\sqrt{2}\end{bmatrix}$$

In this case, the columns of $\mathbf{U}$ are $\mathbf{u}_1 = \begin{bmatrix}3/5\\ 4/5\end{bmatrix}$ and $\mathbf{u}_2 = \begin{bmatrix}-4/5\\ 3/5\end{bmatrix}$, Notice that $\mathbf{u}_i \cdot \mathbf{u}_i = 1$ $(i = 1, 2)$, and that $\mathbf{u}_1 \cdot \mathbf{u}_2 = 0$. This is just another way of saying that the columns are unit vectors and pairwise orthogonal, i.e., are orthonormal. Corresponding properties hold for the columns $\mathbf{v}_1$ and $\mathbf{v}_2$ of $\mathbf{V}$.

Inline Exercise 10.18: What is the first column of $\mathbf{V}$? Be careful that you do not write down the first column of $\mathbf{V}^t$! Verify that the columns of $\mathbf{V}$ are orthonormal. Then verify that the *rows* are as well. This is an example of a general phenomenon: if the columns of a square matrix are orthonormal, then so are the rows.

We'll now show that $\mathbf{M}$ can be written as a sum of outer products,

$$\mathbf{M} = d_1\mathbf{u}_1\mathbf{v}_1^t + d_2\mathbf{u}_1\mathbf{v}_2^t,$$

by direct computation:

$$\begin{aligned}
d_1\mathbf{u}_1\mathbf{v}_1^t + d_2\mathbf{u}_1\mathbf{v}_2^t &= 3\begin{bmatrix}3/5\\ 4/5\end{bmatrix}\begin{bmatrix}1/\sqrt{2} & 1/\sqrt{2}\end{bmatrix} + 2\begin{bmatrix}-4/5\\ 3/5\end{bmatrix}\begin{bmatrix}-1/\sqrt{2} & 1/\sqrt{2}\end{bmatrix}\\
&= \frac{1}{5\sqrt{2}}\left(3\begin{bmatrix}3\\ 4\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix} + 2\begin{bmatrix}-4\\ 3\end{bmatrix}\begin{bmatrix}-1 & 1\end{bmatrix}\right)\\
&= \frac{1}{5\sqrt{2}}\left(3\begin{bmatrix}3 & 3\\ 4 & 4\end{bmatrix} + 2\begin{bmatrix}4 & -4\\ -3 & 3\end{bmatrix}\right)\\
&= \frac{1}{5\sqrt{2}}\left(\begin{bmatrix}9 & 9\\ 12 & 12\end{bmatrix} + \begin{bmatrix}8 & -8\\ -6 & 6\end{bmatrix}\right)\\
&= \frac{1}{5\sqrt{2}}\begin{bmatrix}17 & 1\\ 6 & 18\end{bmatrix}
\end{aligned}$$

This rewriting of $\mathbf{M}$ works in general: $\mathbf{M}$ can always be expressed as a sum of outer products of corresponding columns of $\mathbf{U}$ and $\mathbf{V}$, each multiplied by the corresponding diagonal element of $\mathbf{D}$. This is so useful, in so many areas of graphics, that we restate it as a second form of the SVD:

The Decomposition Theorem: If $\mathbf{M}$ is a $n \times k$ matrix, and $r = \min(n, k)$, and the SVD of $\mathbf{M}$ is

$$\mathbf{M} = \mathbf{UDV}^t,$$

then

$$\mathbf{M} = d_1\mathbf{u}_1\mathbf{v}_1^t + d_2\mathbf{u}_2\mathbf{v}_2^t + \ldots + d_r\mathbf{u}_r\mathbf{v}_r^t,$$

where the $\mathbf{u}_i$ and $\mathbf{v}_i$ are the columns of $\mathbf{U}$ and $\mathbf{V}$, respectively, and the d_i are the diagonal entries of the diagonal matrix $\mathbf{D}$.

The proof is not difficult (see Exercise 1.1.4). It relies on the idea that if $\mathbf{A}$ is an $n \times k$ matrix and $\mathbf{Ax} = \mathbf{Bx}$ for k independent vectors $\mathbf{x}$, then $\mathbf{A} = \mathbf{B}$.

Exercise 10.2: (Uses linear algebra.) Assuming the existence of the SVD for a matrix $\mathbf{M}$, prove the decomposition theorem by showing that (a) The columns of $\mathbf{V}$ form a basis of r-dimensional space, so that any vector $\mathbf{x}$ can be written in the form

$$\mathbf{x} = \alpha_1 \mathbf{v}_1 + \ldots + \alpha_n \mathbf{v}_n,$$

which in turn can be written as $\mathbf{V}\begin{bmatrix}\alpha_1 & \ldots & \alpha_n\end{bmatrix}$, and (b) the products of $\mathbf{UDV}$ with $\mathbf{x}$ (written in this form) and of $d_1\mathbf{u}_1\mathbf{v}_1^t + d_2\mathbf{u}_2\mathbf{v}_2^t + \ldots + d_r\mathbf{u}_r\mathbf{v}_r^t$ with $\mathbf{x}$ are identical.

Programming Exercise 10.1: An image (i.e., an array of grey-scale values between zero and one, say) can be thought of as a large matrix, $\mathbf{M}$ (indeed, this is how we usually represent images in our programs). Use a linear algebra library to compute the SVD $\mathbf{M} = \mathbf{UDV}^t$ of some image $\mathbf{M}$. According to the decomposition theorem, this describes the image as a sum of outer products of many vectors. If we replace the last 90% of the diagonal entries of $\mathbf{D}$ with zeros to get a new matrix $\mathbf{D}'$, then the product $\mathbf{M}' = \mathbf{UD}'\mathbf{V}$ deletes 90% of the terms in this sum of outer products. In doing so, however, it deletes the *smallest* 90% of the terms. Display $\mathbf{M}'$ and compare it to $\mathbf{M}$. Experiment with other values than 90%. At what level do the two images become indistinguishable?

Exercise 10.3: (Uses linear algebra). The *rank* of a matrix is the number of linearly independent columns of the matrix. (a) Explain why the outer product of two vectors always has rank one. (b) The decomposition theorem expresses a matrix $\mathbf{M}$ as a sum of rank one matrices. Explain why the sum of the first p such outer products has rank p (assuming $d_1, d_2, ..., d_p \neq 0$). In fact, this sum $\mathbf{M}_p$ is the rank p matrix that's closest to $\mathbf{M}$, in the sense that the sum of the squares of the entries of $\mathbf{M} - \mathbf{M}_p$ is as small as possible. (You need not prove this.)

10.1.5 The SVD and pseudo-inverses

Again in the special case where $n = k$ so that $\mathbf{U}$ and $\mathbf{V}$ are square, it's easy to compute $\mathbf{M}^{-1}$ if you know the SVD:

$$\mathbf{M}^{-1} = \mathbf{V}D^{-1}\mathbf{U}'$$

where D^{-1} is easy to compute: you simply invert all the elements of the diagonal. If one of these elements is zero, the matrix is singular, and no such inverse exists; in this case, the *pseudoinverse* is also often useful; it'd

defined

$$\mathbf{M}^\dagger = \mathbf{V}D^\dagger\mathbf{U}'$$

where $D^\dagger$ is just D with every non-zero entry inverted (i.e., you try to invert the diagonal matrix D by inverting diagonal elements, and every time you encounter a zero on the diagonal, you ignore it and simply write down zero in the answer). The definition of the pseudoinverse makes sense even when $n \neq k$; the pseudoinverse can be used to solve "least squares" problems, which frequently arise in graphics.

The Pseudoinverse Theorem: (a) If M is an $n \times k$ matrix with $n > k$, the equation $\mathbf{Mx} = \mathbf{b}$ represents an overdetermined system of equations[2] which may have no solution. The vector

$$\mathbf{x}_0 = \mathbf{M}^\dagger\mathbf{b}$$

represents an optimal "solution" to this system, in the sense that $\mathbf{Mx}_0$ is as close to b as possible.

(b) If M is an $n \times k$ matrix with $n < k$, and rank n, the equation $\mathbf{Mx} = \mathbf{b}$ represents an underdetermined system of equations[3]. The vector

$$\mathbf{x}_0 = \mathbf{M}^\dagger\mathbf{b}$$

represents an optimal solution to this system, in the sense that it is the one whose length is smallest.

Here are examples of each of these cases.

Example 1: An overdetermined system. The system

$$\begin{bmatrix}2\\1\end{bmatrix}\begin{bmatrix}x\end{bmatrix} = \begin{bmatrix}4\\3\end{bmatrix}$$

has *no* solution: there's simply no number x with $2x = 4$ and $1x = 3$. But among all the multiples of $\mathbf{M} = \begin{bmatrix}2\\1\end{bmatrix}$, there *is* one that's closest to the vector $\mathbf{b} = \begin{bmatrix}4\\3\end{bmatrix}$, namely $2.2\begin{bmatrix}2\\1\end{bmatrix} = \begin{bmatrix}4.4\\2.2\end{bmatrix}$, as you can discover with elementary geometry. The theorem tells us we can compute this directly, however, using the pseudoinverse. The SVD and pseudoinverse of M are

$$\mathbf{M} = \mathbf{UDV}^t = (\frac{1}{\sqrt{5}}\begin{bmatrix}2\\1\end{bmatrix})\begin{bmatrix}\sqrt{5}\end{bmatrix}\begin{bmatrix}1\end{bmatrix} \tag{10.10}$$

$$\mathbf{M}^\dagger = \mathbf{VD}^\dagger\mathbf{U} = \begin{bmatrix}1\end{bmatrix}\begin{bmatrix}1/\sqrt{5}\end{bmatrix}(\frac{1}{\sqrt{5}}\begin{bmatrix}2 & 1\end{bmatrix} \tag{10.11}$$

$$= \begin{bmatrix}0.4 & 0.2\end{bmatrix} \tag{10.12}$$

[2] I.e., a situation like "five equations in three unknowns."

[3] I.e., a situation like "three equations in five unknowns"

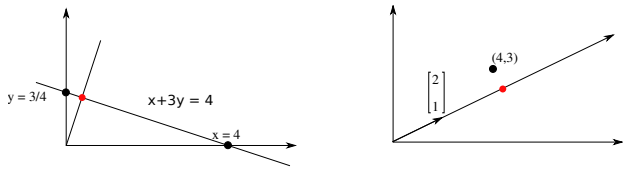

Figure 10.9: (a) The line $x + 3y = 4$ contains many points, so the first example system is underconstrained, and has many solutions. Among these, though, is one (shown in red) that's closest to the origin. (b) The equations $2x = 4$ and $1x = 3$ have no common solution. But the multiples of the vector $[2 \quad 1]^t$ form a line in the plane that passes by the point $(4, 3)$, and there's a point of this line (shown in red) that's as close to $(4, 3)$ as possible.

fig:pseudoinv

And the solution guaranteed by the theorem is

$$\mathbf{M}^\dagger \mathbf{b} = \begin{bmatrix} 0.4 & 0.2 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = 2.2.$$

Example 2: An underdetermined system. The system

$$\begin{bmatrix} 1 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 4$$

has a great many solutions; any point (x, y) on the line $x + 3y = 4$ is a solution. The solution that's *closest to the origin* is the point on the line $x + 3y = 4$ that's as near to $(0, 0)$ as possible, which turns out to be $x = 0.4; y = 1.2$. In this case, the matrix $\mathbf{M}$ is $\begin{bmatrix} 1 & 3 \end{bmatrix}$; its SVD and pseudoinverse are simply

$$\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{V}^t = \begin{bmatrix} 1 \end{bmatrix} \begin{bmatrix} \sqrt{10} \end{bmatrix} \begin{bmatrix} 1/\sqrt{10} & 3/\sqrt{10} \end{bmatrix}, \text{and} \tag{10.13}$$

$$\mathbf{M}^\dagger = \mathbf{V}\mathbf{D}^\dagger\mathbf{U} = \begin{bmatrix} 1/\sqrt{10} \\ 3/\sqrt{10} \end{bmatrix} \begin{bmatrix} 1/\sqrt{10} \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} = \begin{bmatrix} 1/10 \\ 3/10 \end{bmatrix}. \tag{10.14}$$

And the solution guaranteed by the theorem is

$$\mathbf{M}^\dagger \mathbf{b} = \begin{bmatrix} 1/10 \\ 3/10 \end{bmatrix} \begin{bmatrix} 4 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 1.2 \end{bmatrix}.$$

Of course, this kind of computation is much more interesting in the case where the matrices are much larger, but all the essential characteristics are present even in these simple examples.

A particularly interesting example arises when one has, for instance, two polyhedral models (consisting of perhaps hundreds of vertexes joined by triangular faces) that might be "essentially identical": one might be just a translated and rotated and scaled version of the other. We'll see, in section **??**, how to represent translation along with rotation and scaling in

terms of matrix multiplication. With this, we can determine whether the two models are in fact essentially identical by listing the coordinates of the first in the columns of a matrix $\mathbf{V}$, and the coordinates of the second in a matrix $\mathbf{W}$, and then seeking a matrix $\mathbf{A}$ with

$$\mathbf{AV} = \mathbf{W}.$$

This amounts to solving the "overconstrained system" problem many times; we find that $\mathbf{A} = \mathbf{V}^\dagger\mathbf{W}$ is the best possible solution. If, having computed this, we find that

$$\mathbf{AV} = \mathbf{W},$$

then the models are essentially identical; if the left and right sides differ, then the models are not essentially identical. (This entire approach depends, of course, on corresponding vertexes of the two models being listed in the corresponding order; the more general problem is quite a lot more difficult.)

10.1.6 Computing the SVD

If you're wondering to yourself "How would I ever find $\mathbf{U}$, $\mathbf{D}$ and $\mathbf{V}$?" you're asking the right questions. In general it's relatively difficult, and we rely on numerical linear algebra packages to do it for us. Furthermore, the results are by no means unique: a single matrix may have multiple singular-value decompositions. For instance, if $\mathbf{S}$ is *any* $n \times n$ matrix with orthonormal columns, then

$$\mathbf{I} = \mathbf{SIS}^t$$

is one possible singular-value decomposition of the identity matrix. Even though there are many possible SVDs, the singular values are the same for all decompositions.

The *rank* of the matrix $\mathbf{M}$, which is defined as the number of linearly independent columns, turns out to be exactly the number of nonzero singular values.

10.1.7 The SVDs of our example transformations

Let's apply the SVD to some of our example transformations. First, note that the matrix $\mathbf{U}$ will have columns $\mathbf{u}_1$ and $\mathbf{u}_2$ that are unit vectors in $\mathbf{R^2}$, which means they have to have the form

$$\begin{bmatrix}\cos\theta\\ \sin\theta\end{bmatrix}$$

for some $\theta \in \mathbf{R}$. Since they are perpendicular, $\mathbf{u}_2$ must be $\pi/2$ clockwise or counterclockwise from $\mathbf{u}_1$; if it's clockwise, we can negate both it and $d_{2,2}$;

we can do the same for the columns of $\mathbf{V}$. So we can actually use the SVD to write each matrix $\mathbf{M}$ in the form

$$\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{V}^t,$$

where $\mathbf{U}$ and $\mathbf{V}$ are both rotation matrices (like the one in T_1), and $\mathbf{D}$ is diagonal.

One possible decomposition for the rotation matrix $\mathbf{M}_1$ of T_1 is simply

$$\mathbf{M}_1 = \mathbf{I}\mathbf{I}\mathbf{M}_1.$$

So in the case of rotations, the SVD doesn't tell us much.

For the scaling matrix M_2 of T_2, we get a similarly simple decomposition:

$$\mathbf{M}_2 = \mathbf{I}\mathbf{M}_2\mathbf{I}.$$

The shearing matrix of T_3 is somewhat more interesting. One decomposition of it is

$$\begin{bmatrix}1 & 2\\ 0 & 1\end{bmatrix} = \begin{bmatrix}\cos 22.5^\circ & -\sin 22.5^\circ\\ \sin 22.5^\circ & \cos 22.5^\circ\end{bmatrix}\begin{bmatrix}\sqrt{2}+1 & 0\\ 0 & \sqrt{2}-1\end{bmatrix}\begin{bmatrix}\cos 67.5^\circ & -\sin 67.5^\circ\\ \sin 67.5^\circ & \cos 67.5^\circ\end{bmatrix}^t$$

where the simplicity of the angles is a consequence of the small integer entries of the matrix. If we redraw this transformation (see figure 1.10) with new axes (the first set, in the domain, rotated 67.5° from the usual coordinate axes, and the second set, in the codomain, rotated 22.5° from the usual axes), we can see that the transformation amounts to a stretching by about 2.4 along the first coordinate, and a scaling by about 0.41 along the second, just as indicated by the diagonal matrix. We've also drawn a box, aligned with the tilted axes, and the transformed box, to make this easier to see.

The general transformation, T_4, has an SVD which to two decimal places is

$$\begin{bmatrix}1 & -1\\ 2 & 2\end{bmatrix} = \begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}\begin{bmatrix}2.82 & 0\\ 0 & 1.41\end{bmatrix}\begin{bmatrix}-.71 & -.71\\ 0.71 & -0.71\end{bmatrix}.$$

You should draw the coordinate axes implied by this decomposition on the picture of T_4 to verify that it really does appear, in those coordinates, to stretch the first axis by a factor of 2.82, and to stretch the second axis by 1.41.

In general, if two of the singular values (the entries of $\mathbf{D}$) are equal, then the SVD is not unique. If all are distinct, and we require that they appear in decreasing order ($d_1 > d_2 > \ldots$), then $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}$ are in fact unique (up to negations of corresponding columns of $\mathbf{U}$ and $\mathbf{V}$).

Explicitly computing the singular-value decomposition of a matrix efficiently is a topic beyond this book, but efficient implementations are widely available.

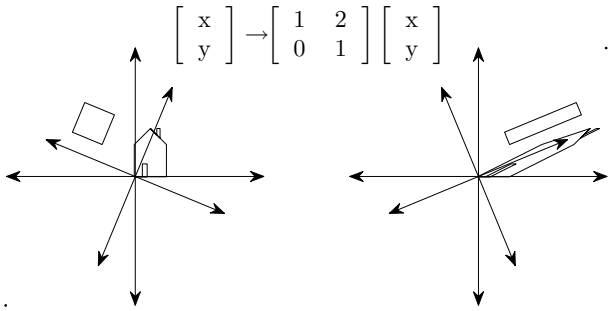

Figure 10.10: The transformation T_2 again, this time with a new set of axes drawn in the domain and codomain, and a second object (a box, aligned with the new axes) being transformed along with the house. We can see that in these new coordinates, our shearing transformation is just an expansion along the first axis and compression along the second.

fig:svd-example2

Exercise 10.4: Suppose that $T : \mathbf{R^2} \to \mathbf{R^2}$ is a linear transformation represented by the 2×2 matrix M, i.e. $T(\mathbf{v}) = \mathbf{Mv}$. Let $K = \max_{\mathbf{v} \in \mathbf{S^1}} \|T(\mathbf{v})\|$, i.e., K is the maximum length of all unit vectors transformed by M. (a) If the SVD of M is $\mathbf{M} = \mathbf{UDV}^t$, show that $K = d_1^2$. (b) What is the *minimum* length among all such vectors, in terms of D? (c) Generalize to $\mathbf{R^3}$.

10.2 Translation

We now describe a way to apply linear transformations to generate *translations*, and at the same time give a nice model for the points-versus-vectors ideas we've espoused so far.

The idea is this: as our Euclidean plane (i.e., our set of *points*), we'll take the plane $w = 1$ in xyw-space (see Figure 1.11). The use of w here is in preparation for what we'll do in 3-space, which is to consider the 3-dimensional set defined by $w = 1$ in $xyzw$-space. .

Having done this, we can consider transformations that multiply such vectors by a 3×3 matrix M. The only problem is that the result of such a multiplication may not have a 1 as its last entry. We can restrict our attention to those that do:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ p & q & r \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}.$$

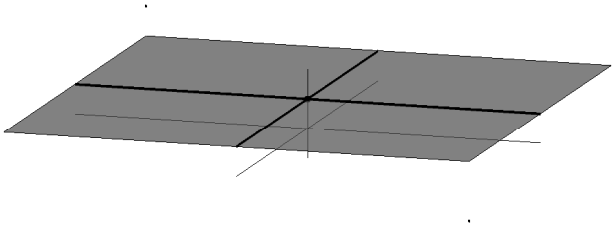

Figure 10.11: The $w = 1$ plane in xyw-space will serve as our set of points as we discuss 2D transformations that can now include translation. We'll find transformations from $\mathbf{R^3}$ to $\mathbf{R^3}$ that send points with $w = 1$ to new points with $w = 1$, thus effectively giving us a mapping from the Euclidean plane to itself.

fig:homog-space

For this equation to hold for every x and y, we must have $px + qy + r = 1$ for all x, y. This forces $p = q = 0$ and $r = 1$.

Thus we'll consider transformations of the form

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}.$$

If we examine the special case where the upper-left corner is a 2×2 identity matrix, we get

$$\begin{bmatrix} 1 & 0 & c \\ 0 & 1 & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + c \\ y + f \\ 1 \end{bmatrix}.$$

As long as we pay attention only to the x and y coordinates, this looks like a translation! We've added c to each x-coordinate and f to each y-coordinate (see figure 1.12).

On the other hand, if we make $c = f = 0$, then the third coordinate becomes irrelevant, and the upper-left 2×2 matrix can perform any of the operations we've seen up until now. Thus with the simple trick of adding a third coordinate, and requiring that it always be one, we've managed to unify rotation, scaling, and all the other linear transformations with the new class of transformations, *translations*.

10.3 Points and Vectors Again

Back in chapter 7, we said that points and vectors could be combined in certain ways: the difference of points is a vector, a vector could be added to a point to get a new point, and more generally, affine combinations of

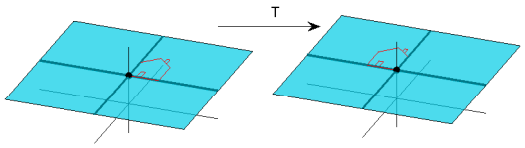

Figure 10.12: A *linear* transformation on xyw-space whose matrix has last row $[0\ 0\ 1]$ sends points in the $w = 1$ plane to other points in the $w = 1$ plane linearly, but it can also *translate* them within the plane. In this figure, all points of the $w = 1$ are translated one unit to the left.

points, that is, combinations of the form

$$\alpha_1 P_1 + \alpha_2 P_2 + \ldots + \alpha_k P_k, \tag{10.15}$$

were allowed if and only if $\alpha_1 + \alpha_2 + \ldots + \alpha_k = 1$.

We now have a situation in which these distinctions makes sense in terms of familiar mathematics: we can regard *points* of the plane as being elements of $\mathbf{R^3}$ whose third coordinate is one, and *vectors* as being elements of $\mathbf{R^3}$ whose third coordinate is zero.

With this convention, it's clear that the difference of points is a vector, the sum of a vector and a point is a point, and that combinations like the one in Equation 1.15 yield a point if and only if the sum of the coefficients is one (because the third coordinate of the result will be exactly the sum of the coefficients; for the sum to be a *point*, this third coordinate is required to be one).

You may ask "Why, when we're already familiar with vectors in 3-space, should we bother calling some of them "points in the Euclidean plane" and others "two-dimensional vectors"?" The answer is that the distinctions have geometric significance when we're using this subset of 3-space as a model for 2D transformations. Adding vectors in 3-space is defined in linear algebra, but adding together two of our "points" gives a location in 3-space that's not on the $w = 1$ plane or the $w = 0$ plane, so we don't have a name for it at all.

Henceforth we'll use E^2 (for "Euclidean 2-dimensional space") to denote this $w = 1$ plane in xyw-space, and write (x, y) to mean the point of E^2

corresponding to the 3-space vector $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$.

10.4 Why use 3×3 matrices instead of a matrix and a vector?

Students sometimes wonder "why not just represent a linear-transformation-plus-translation in the form

$$T(\mathbf{x}) = \mathbf{M}\mathbf{x} + \mathbf{b},$$

where the matrix M represents the linear stuff (rotating, scaling, shearing) and b represents the translation?"

First, you *can* do that, and it works just fine. You might save a tiny bit of storage (four numbers for the matrix, two for the vector, so six numbers instead of nine), but since our matrices always have two zeroes and a one in the third column, we don't really need to store that column anyhow, so it's the same). Otherwise, there's no important difference.

Second, the reason to unify the transformations into a single matrix is that it's then very easy to take multiple transformations (each represented by a matrix) and compose them (i.e., perform one after another): we just multiply their matrices together in the right order to get the matrix for the composed transformation. You can do this in the matrix-and-vector formulation as well, but the programming is slightly messier, and more error-prone.

There's a third reason, however: it'll soon become apparent that we can also work with triples whose third entry is neither one nor zero, and the operation of "homogenization" (dividing by w) to get points (except when $w = 0$) to get yet more transformations, this time getting transformations that are related to what happens when we view an object in perspective.

10.4.1 Decomposition of transformations

An important consequence of the singular value decomposition, and the "extra coordinate" representation of translation, is that every transformation matrix can be written as a specific kind of product: a translation, followed by a rotation, a scale, and another rotation. That means that no matter what sequence of operations you apply: two translations, a rotate, another translate, a scale, another rotate, another translate, and two more scales, the resulting sequence can always be expressed as a translate, then rotate, then scale, then rotate. We'll explain how to find this factorization presently.

There is nothing particularly special about this order of operations: we could do the translation last instead of first; the sequence of operations is

also not unique. Neither of these matters a great deal, however, in the main application of the ideas: if we've written a matrix $\mathbf{M}$ in the form

$$\mathbf{M} = \mathbf{R}_2\mathbf{S}\mathbf{R}_1\mathbf{T},$$

where $\mathbf{R}_1$ and $\mathbf{R}_2$ are rotations, $\mathbf{S}$ is a scale, and $\mathbf{T}$ is a translation, then it's easy to invert $\mathbf{M}$, because the inverse of a rotation matrix is its transpose; the inverse of a scale matrix is gotten by inverting each diagonal element, and the inverse of a translation by $\mathbf{v}$ is translation by $-\mathbf{v}$. So if

$$\mathbf{M} = \begin{bmatrix} \cos 30 & -\sin 30 & 0 \\ \sin 30 & \cos 30 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 70 & -\sin 70 & 0 \\ \sin 70 & \cos 70 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{bmatrix},$$

then

$$\mathbf{M}^{-1} = \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & -6 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} \cos 70 & \sin 70 & 0 \\ -\sin 70 & \cos 70 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & /13 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 30 & \sin 30 & 0 \\ -\sin 30 & \cos 30 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Inline Exercise 10.19: Explain why the order of the matrices expressing $\mathbf{M}$ and $\mathbf{M}^{-1}$ is reversed (e.g., why the translation moves from the last term of the product to the first term of the product).

The recipe to actually compute the decomposition of a matrix representing an arbitrary sequence of rotations, translations, and scales into the translate-rotate-scale-rotate form is quite simple. First, we note that the bottom row of such a matrix is always $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$, so the matrix has the form

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}.$$

We let $\mathbf{T} = \begin{bmatrix} 1 & 0 & c \\ 0 & 1 & f \\ 0 & 0 & 1 \end{bmatrix}$; it's easy to verify that

$$\mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 & -c \\ 0 & 1 & -f \\ 0 & 0 & 1 \end{bmatrix}, \text{ and} \tag{10.16}$$

$$\mathbf{M}\mathbf{T}^{-1} = \begin{bmatrix} a & b & 0 \\ d & e & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{10.17}$$

Letting $\mathbf{K}$ be the upper left 2×2 matrix $\begin{bmatrix} a & b \\ d & e \end{bmatrix}$, we compute the SVD of $\mathbf{K}$, i.e., we write

$$\mathbf{K} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathbf{T}}.$$

Then letting $\mathbf{R}_2$ be the matrix whose upper-left 2×2 block is $\mathbf{U}$ (and whose lower right entry is a 1), i.e.,

$$\mathbf{R}_2 = \begin{bmatrix} u_{11} & u_{12} & 0 \\ u_{21} & u_{22} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

We similarly let $\mathbf{R}_1$ be the matrix whose upper left corner is $\mathbf{V^T}$, and fill in the other entries as we did with $\mathbf{R}_2$. And finally, we let $\mathbf{S} = \mathbf{D}$. Then

$$\mathbf{M} = \mathbf{R}_2\mathbf{S}\mathbf{R}_1\mathbf{T},$$

where the $\mathbf{R}$s are rotations, $\mathbf{S}$ is a scale, and $\mathbf{T}$ is a translation.

In the quite common event that $\mathbf{M}$ is known to be a product of a sequence of translations, rotations, and *uniform* scaling operations, the resulting matrix S is a multiple of the identity, and therefore $\mathbf{R}_2\mathbf{S} = \mathbf{S}\mathbf{R}_2$, so the decomposition can be rewritten

$$\mathbf{M} = \mathbf{S}(\mathbf{R}_2\mathbf{R}_1)\mathbf{T},$$

where $\mathbf{R}_2\mathbf{R}_1$ can be multiplied out to yield a single rotation. Hence every sequence of rotations, translations, and uniform scales is equivalent to a single translation followed by a single rotation followed by a single scale.

10.5 Windowing transformations

As an application of our new, richer, set of transformations, let's examine *windowing transformations*, which send one axis-aligned rectangle to another as shown in Figure 13.2. (We already discussed this briefly in Chapter 3.)

We'll first take the direct approach: do a little algebra; then we'll examine a more automated approach.

We'll need to do essentially the same thing to the first and second coordinates, so let's look at how to transform the first coordinate only. We need to send u_1 to x_1 and u_2 to x_2. That means that we need to scale up any coordinate *difference* by the factor $\frac{x_2-x_1}{u_2-u_1}$. So our transformation (for the first coordinate) has the form

$$t \mapsto \frac{x_2 - x_1}{u_2 - u_1}t + \text{something}.$$

If we apply this to $t = u_1$, we know that we want to get x_1; this leads to the equation

$$\frac{x_2 - x_1}{u_2 - u_1}u_1 + \text{something} = x_1;$$

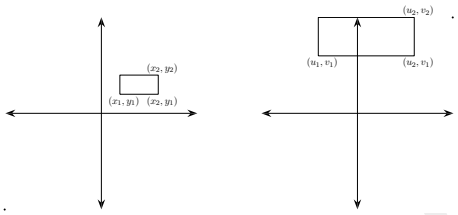

Figure 10.13: We'd like to find a transformation to send the rectangle whose lower left and upper right corners are (u_1, v_1) and (u_2, v_2) to another rectangle whose corresponding corners are (x_1, y_1) and (x_2, y_2).

fig:windowing-xform

Solving for the missing offset gives

$$x_1 - \frac{x_2 - x_1}{u_2 - u_1}u_1 = x_1\frac{u_2 - u_1}{u_2 - u_1} - \frac{x_2 - x_1}{u_2 - u_1}u_1 \tag{10.18}$$

$$= \frac{x_1u_2 - x_1u_1}{u_2 - u_1} - \frac{x_2u_1 - x_1u_1}{u_2 - u_1} \tag{10.19}$$

$$= \frac{x_1u_2 - x_1u_1 - x_2u_1 + x_1u_1}{u_2 - u_1} \tag{10.20}$$

$$= \frac{x_1u_2 - x_2u_1}{u_2 - u_1} \tag{10.21}$$

$$\tag{10.22}$$

$$t \mapsto \frac{x_2 - x_1}{u_2 - u_1}t + \frac{x_1u_2 - x_2u_1}{u_2 - u_1}.$$

Doing essentially the same thing for the v and y terms (i.e., the second coordinate) we get the transformation, which we can write in matrix form:

$$T(\mathbf{x}) = \mathbf{M}\mathbf{x},$$

where

$$\mathbf{M} = \begin{bmatrix} \frac{x_2-x_1}{u_2-u_1} & 0 & \frac{x_1u_2-x_2u_1}{u_2-u_1} \\ 0 & \frac{y_2-y_1}{v_2-v_1} & \frac{y_1v_2-y_2v_1}{v_2-v_1} \\ 0 & 0 & 1 \end{bmatrix}. \tag{10.23}$$

Inline Exercise 10.20: Multiply the matrix M of equation 1.23 by the vector $\begin{bmatrix} u_1, v_1, 1 \end{bmatrix}^t$ to confirm that you do get $\begin{bmatrix} x_1, y_1, 1 \end{bmatrix}^t$. Do the same for the opposite corner of the rectangle.

We'll now show you a second way to build this transformation (and many others as well).

10.6 Building 3D transformations

Just as in $2D$, we could send the vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ to the vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ by building a matrix M whose columns were $\mathbf{v}_1$ and $\mathbf{v}_2$, and then used two such matrices (inverting one along the way) to send any two independent vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ to any two vectors $\mathbf{w}_1$ and $\mathbf{w}_2$, we can do exactly the analogous thing in 3-space: we can send the standard basis vectors $\mathbf{e}_1, \mathbf{e}_2$, and $\mathbf{e}_3$ to any three other vectors, just by using them as the columns of a matrix. Let's start by sending $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ to three corners of our first rectangle — the two we've already specified, and the lower-right one, at location (u_2, v_1). The three vectors corresponding to these points are

$$\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}, \quad \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix}, \text{ and } \begin{bmatrix} u_2 \\ v_1 \\ 1 \end{bmatrix}.$$

Because the three corners of the rectangle are not collinear, the three vectors are independent. So the matrix

$$\mathbf{M}_1 = \begin{bmatrix} u_1 & u_2 & u_2 \\ v_1 & v_2 & v_1 \\ 1 & 1 & 1 \end{bmatrix},$$

which performs the desired transformation, will be invertible.

Exercise 10.5: (Requires linear algebra). Show that three distinct points $P, Q,$ and R in the Euclidean plane are collinear if and only if the corresponding vectors ($\mathbf{v}_P = \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$, etc.) are linearly dependent by showing that if $\alpha_P \mathbf{v}_P + \alpha_Q \mathbf{v}_Q + \alpha_R \mathbf{v}_R = 0$ with not all the α being zero, then (a) *none* of the αs are zero, and (b) the point Q is an affine combination of P and R; in particular, $Q = \frac{\alpha_P}{\alpha_Q} P + \frac{\alpha_R}{\alpha_P} R$, so Q must lie on the line between P and R. (c) Argue why dependence and collinearity are trivially the same if two or more of the points P, Q, and R are identical.

We can similarly build the matrix $\mathbf{M}_2$, with the corresponding xs and ys in it. Finally, we can compute

$$\mathbf{M}_2 \mathbf{M}_1^{-1},$$

which will perform the desired transformation. For instance, the lower-left corner of the starting rectangle will be sent, by $\mathbf{M}_1^{-1}$, to $\mathbf{e}_1$ (because $\mathbf{M}_1$ sent $\mathbf{e}_1$ to the lower-left corner); multiplying $\mathbf{e}_1$ by $\mathbf{M}_2$ will send it to the lower-left corner of the target rectangle. A similar argument applies to all three corners. Indeed, if we compute the inverse algebraically and multiply out everything, we'll once again arrive at the matrix given in equation **??**. But we don't need to do so: we know that this must be the right matrix. Assuming we're willing to use a matrix-inversion routine, there's no need to think through anything more than "I want these three points to be sent to these three other points."

Summary: Given any three non-collinear points P_1, P_2, P_3 in E^2, we can find a matrix transformation send them to any three points Q_1, Q_2, Q_3 with the procedure above.

Programming Exercise 10.2: Use the 2D test harness to write a program to demonstrate windowing transforms. The user should click-and-drag two rectangles, and you should compute the transform between them. Subsequent clicks by the user within the first rectangle should be shown as small dots, and the corresponding locations in the second rectangle should also be shown as dots. Provide a "Clear" button to let the user start over.

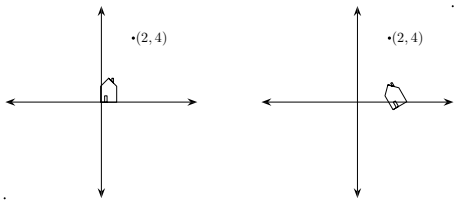

Figure 10.14: We'd like to rotate the entire plane by 30 degrees counterclockwise about the point $P = (2, 4)$.

fig:plane-rot

10.7 Another example of building a 2D transformation

Suppose we want to find a 3×3 matrix transformation that rotates the entire plane 30 degrees counterclockwise around the point $P = (2, 4)$, as shown in Figure 1.14.

Here are two approaches.

First, we know how to rotate about the origin by thirty degrees; we can use the transformation T_1 from the start of the chapter. So we can do our desired transformation in three steps (see Figure 1.15

- Move the point $(2, 4)$ to the origin.
- Rotate by thirty degrees.

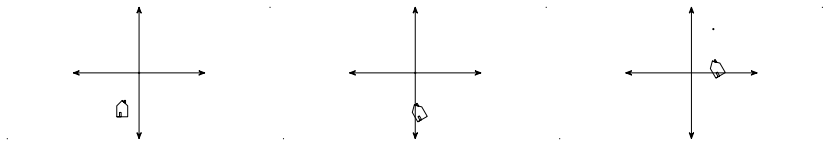

Figure 10.15: Rotating the house about $(2, 4)$ in three steps. (a) We translate everything so that the point $(2, 4)$ moves to the origin. (b) We rotate about the origin by 30 degrees. (c) We translate the origin back to $(2, 4)$.

fig:transform-steps

- Move the origin back to $(2, 4)$.

The matrix that moves the point $(2, 4)$ to the origin is

$$\begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}.$$

The one that moves it back is similar, except that the 2 and 4 are not negated. And the rotation matrix (expressed in our new 3×3 format) is

$$\begin{bmatrix} \cos 30^\circ & -\sin 30^\circ & 0 \\ \sin 30^\circ & \cos 30^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The matrix representing the entire sequence of transformations is therefore

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 30^\circ & -\sin 30^\circ & 0 \\ \sin 30^\circ & \cos 30^\circ & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}.$$

Inline Exercise 10.21: (a) Explain why this is the correct order in which to multiply the transformations to get the desired result. (b) Verify that the point $(2, 4)$ is indeed left unmoved by multiplying $\begin{bmatrix} 2 & 4 & 1 \end{bmatrix}^t$ by the sequence of matrices above.

The second approach is again more automatic: we find three points whose target locations we know, just as we did with the windowing transformation above. We'll use $P = (2, 4)$, $Q = (3, 4)$, the point that's one unit to the right of P, and $R = (2, 5)$, the point one unit above P. We know that we want P sent to P, Q sent to $(2 + \cos 30^\circ, 4 + \sin 30^\circ)$, and R sent to $(2 - \sin 30^\circ, 4 + \cos 30^\circ)$. (Draw a picture to convince yourself that these are correct). The matrix that achieves this is just

$$\begin{bmatrix} 2 & 2 + \cos 30^\circ & 4 - \sin 30^\circ \\ 4 & 4 + \sin 30^\circ & 4 + \cos 30^\circ \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 & 2 \\ 4 & 4 & 5 \\ 1 & 1 & 1 \end{bmatrix}^{-1}.$$

Both approaches are reasonably easy to work with.

There's a third approach – a variation of the second – in which we specify where we want to send a point and two vectors, rather than three points. In this case, we might say that we want the point P to remain fixed, and the vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ to go to

$$\begin{bmatrix}\cos 30^\circ \\ \sin 30^\circ \\ 0\end{bmatrix} \text{ and } \begin{bmatrix}-\sin 30^\circ \\ \cos 30^\circ \\ 0\end{bmatrix},$$

respectively. In this case, instead of finding matrices that send the vectors $\mathbf{e}_1, \mathbf{e}_2$, and $\mathbf{e}_3$ to the desired three points, before and after, we find matrices that send those vectors to the desired point-and-two-vectors, before and after. These matrices are

$$\begin{bmatrix}2 & 1 & 0 \\ 4 & 0 & 1 \\ 1 & 0 & 0\end{bmatrix} \text{ and } \begin{bmatrix}2 & \cos 30^\circ & -\sin 30^\circ \\ 4 & \sin 30^\circ & \cos 30^\circ \\ 1 & 0 & 0\end{bmatrix}$$

so the overall matrix is

$$\begin{bmatrix}2 & \cos 30^\circ & -\sin 30^\circ \\ 4 & \sin 30^\circ & \cos 30^\circ \\ 1 & 0 & 0\end{bmatrix} \begin{bmatrix}2 & 1 & 0 \\ 4 & 0 & 1 \\ 1 & 0 & 0\end{bmatrix}^{-1}.$$

These general techniques can be applied to create any linear-plus-translation transformation of the $w = 1$ plane, but there are some specific ones that are good to know. Rotation in the xy-plane, by an amount θ (rotating the positive x-axis towards the positive y-axis) is given by

$$R_{xy}(\theta) = \begin{bmatrix}\cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1\end{bmatrix}.$$

In some books and software packages, this is called "rotation around z;" we prefer the term "rotation in the xy-plane" because it also indicates the direction of rotation (from x, towards y). The other two standard rotations are

$$R_{yz}(\theta) = \begin{bmatrix}1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta\end{bmatrix},$$

and

$$R_{zx}(\theta) = \begin{bmatrix}\cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta\end{bmatrix};$$

note that the last expression rotates z towards x, and *not* the opposite. Using this naming convention helps keep the pattern of plusses and minuses symmetric.

10.8 Transforming vectors and covectors

We've agreed to say that the point $(x, y) \in E^2$ corresponds to the 3-space vector $\begin{bmatrix} x & y & 1 \end{bmatrix}^t$, and that the vector $\begin{bmatrix} u \\ v \end{bmatrix}$ corresponds to the 3-space vector $\begin{bmatrix} u & v & 0 \end{bmatrix}^t$. If we use a 3×3 matrix M (with last row $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$) to transform 3-space, via

$$T : \mathbf{R^3} \to \mathbf{R^3} : \mathbf{x} \mapsto \mathbf{Mx},$$

then the restriction of T to the $w = 1$ plane has its image in E^2 as well, so we can write

$$(T|E^2) : E^2 \to E^2 : \mathbf{x} \mapsto \mathbf{Mx},$$

But we also noted above that we could regard T as transforming *vectors*, i.e., displacements of two-dimensional Euclidean space, which are typically written with two coordinates, but which we represent in the form $\begin{bmatrix} u & v & 0 \end{bmatrix}^t$. Because the last entry of such a "vector" is always zero, the last column of M has no effect on how vectors are transformed. Instead of computing

$$\mathbf{M} \begin{bmatrix} u \\ v \\ 0 \end{bmatrix},$$

we could equally well compute

$$\begin{bmatrix} m_{1,1} & m_{1,2} & 0 \\ m_{2,1} & m_{2,2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ 0 \end{bmatrix},$$

and the result would have a zero in the third place. In fact, we could transform such vectors directly as two-coordinate vectors, by simply computing

$$\begin{bmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}.$$

For this reason, it's sometimes said for an affine transformation of the Euclidean plane represented by multiplication by a matrix M, the associated transformation of vectors is represented by

$$\overline{\mathbf{M}} = \begin{bmatrix} m_{1,1} & m_{1,2} \\ m_{2,1} & m_{2,2} \end{bmatrix}.$$

What about covectors? Recall that a typical covector could be written in the form

$$\phi_{\mathbf{w}} : \mathbf{R}^2 \to \mathbf{R}^2 : \mathbf{v} \mapsto \mathbf{w} \cdot \mathbf{v},$$

where w was some vector in $\mathbf{R}^2$. We'd like to transform $\phi_{\mathbf{w}}$ in a way that's consistent with T. Figure 1.16

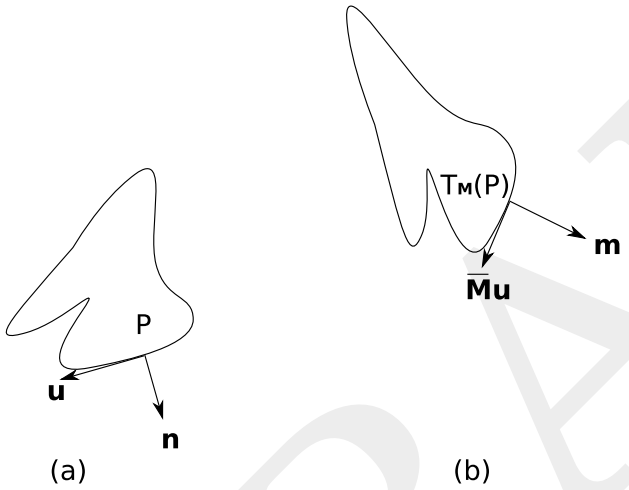

Figure 10.16: (a) A geometric shape that's been modeled using some modeling tool; the normal vector $\mathbf{n}$ at a particular point P has been computed too. The vector $\mathbf{u}$ is tangent to the shape at P. (b) The shape has been translated, rotated, and scaled as it was placed into a scene. At the transformed location of P, we want to find the normal vector $\mathbf{m}$ with the property that its inner product with the transformed tangent $\overline{\mathrm{M}}\mathbf{u}$ is still zero.

fig:normal-xform

XXX shows why: we often build a geometric model of some shape, and compute all the normal vectors to the shape. Suppose that $\mathbf{n}$ is one such surface normal. We then place that shape into 3-space by applying some "modeling transformation" T_{M} to it, and we'd like to know the normal vectors to that transformed shape, so that we can do things like compute the angle between a light-ray $\mathbf{v}$ and that surface normal. If we call the transformed surface normal $\mathbf{m}$, we want to compute $\mathbf{v} \cdot \mathbf{m}$. How is $\mathbf{m}$ related to the surface normal $\mathbf{n}$ of the original model?

The original surface normal $\mathbf{n}$ was defined by the property that it was orthogonal to every vector $\mathbf{u}$ that was tangent to the surface. The new normal vector $\mathbf{m}$ must be orthogonal to all the transformed tangent vectors, which are tangent to the transformed surface, i.e., we need to have

$$\mathbf{m} \cdot \overline{\mathbf{M}}\mathbf{u} = 0$$

for every tangent vector $\mathbf{u}$ to the surface. In fact, we can go further: for *any* vector $\mathbf{u}$, we'd like to have

$$\mathbf{m} \cdot \overline{\mathbf{M}}\mathbf{u} = \mathbf{n} \cdot \mathbf{u},$$

that is, we'd like to be sure that the angle between an untransformed vector and $\mathbf{n}$ is the same as the angle between a transformed vector and $\mathbf{m}$.

FIGURE HERE

Before working through this, let's look at a couple of examples: In the case of the transformation T_1, the vector perpendicular to the bottom side of the house (we'll use this as our vector $\mathbf{n}$) should be transformed so that it's still perpendicular to the bottom of the transformed house. This is achieved by rotating it by 30 degrees (see figure 1.17).

If we just *translate* the house, then $\mathbf{n}$ again should be transformed just the way we transform ordinary vectors, i.e., not at all.

But what about when we shear the house, as with example transformation T_3? The associated vector transformation is still a shearing transformation; it takes a vertical vector and tilts it! But the vector $\mathbf{n}$, if it's to remain perpendicular to the bottom of the house, must not be changed at all. So in this case we see the necessity of transforming covectors differently from vectors.

Let's write down, once again, what we want. We're looking for a vector $\mathbf{m}$ that satisfies

$$\mathbf{m} \cdot (\overline{\mathbf{M}}\mathbf{u}) = \mathbf{n} \cdot \mathbf{u}$$

for every possible vector $\mathbf{v}$. To make the algebra more obvious, let's swap the order of the vectors, and say that we want

$$(\overline{\mathbf{M}}\mathbf{u}) \cdot \mathbf{m} = \mathbf{u} \cdot \mathbf{n}$$

Recalling that $\mathbf{a} \cdot \mathbf{b}$ can be written $\mathbf{a}^t\mathbf{b}$, we can rewrite this as

$$(\overline{\mathbf{M}}\mathbf{u})^t\mathbf{m} = \mathbf{u}^t\mathbf{n}.$$

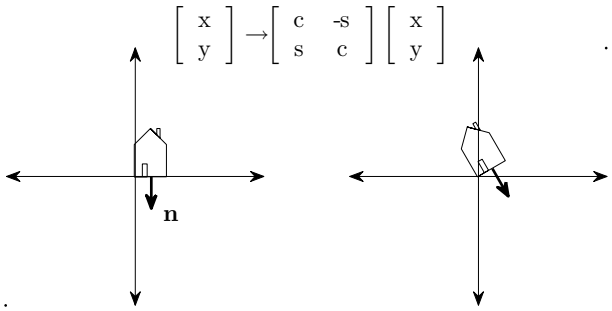

Figure 10.17: The vector $\mathbf{n}$ is perpendicular to the bottom of the house. If we want to measure the angle between other vectors (like one pointing horizontally along the bottom of the house) and $\mathbf{n}$, and we want to do so either before or after transformation, then $\mathbf{n}$ must be transformed as well. In this case, rotating $\mathbf{n}$ by the same 30 degrees that we used to rotate the house works fine.

fig:covec-xform

Remembering that $(\mathbf{AB})^t = \mathbf{A}^t\mathbf{B}^t$, and then simplifying, we get

$$(\overline{\mathbf{M}}\mathbf{u})^t\mathbf{m} = \mathbf{u}^t\mathbf{n}$$
$$(\mathbf{u}^t\overline{\mathbf{M}}^t)\mathbf{m} = \mathbf{u}^t\mathbf{n}$$
$$\mathbf{u}^t(\overline{\mathbf{M}}^t\mathbf{m}) = \mathbf{u}^t\mathbf{n}$$

where the last step follows from the associativity of matrix multiplication. This last equality is of the form $\mathbf{u}\cdot\mathbf{a} = \mathbf{u}\cdot\mathbf{b}$ for all $\mathbf{u}$. Such an equality holds if and only if $\mathbf{a} = \mathbf{b}$, i.e., if and only if

$$\overline{\mathbf{M}}^t\mathbf{m} = \mathbf{n},$$

i.e.,

$$\mathbf{m} = (\overline{\mathbf{M}}^t)^{-1}\mathbf{n},$$

where we are assuming that $\overline{\mathbf{M}}$ is invertible.

We can therefore conclude that the covector $\phi_{\mathbf{n}}$ transforms to the covector $\phi_{(\overline{\mathbf{M}}^t)^{-1}\mathbf{n}}$. For this reason, the inverse-transpose is sometimes referred to as the *covector transformation* or (because of its frequent application to normal vectors), the *normal transform*. Note that if we choose to write covectors as row-vectors, then the transpose is not needed, but that we have to multiply the row-vector on the *right* by $\overline{\mathbf{M}}^{-1}$.

Taking our shearing transformation, T_3, as an example, when written in xyw-space the matrix M for the transformation is

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

and hence $\overline{\mathbf{M}}$ is

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

while the "normal transform" is

$$(\overline{\mathbf{M}}^{-1})^t = \begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix}^t == \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix}.$$

Hence the covector $\phi_{\mathbf{n}}$, where $\mathbf{n} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ becomes the covector $\phi_{\mathbf{m}}$, where $\mathbf{m} = \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix} \mathbf{n} = \begin{bmatrix} 3 \\ -3 \end{bmatrix}$.

Inline Exercise 10.22: (a) Find an equation (in coordinates, not vector form) for a line passing through the point $P = (1, 1)$, with normal vector $\mathbf{n} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$. (b) Find a second point Q on this line. (c) Find $P' = T_3(P)$ and $Q' = T_3(Q)$, and a coordinate equation of the line joining P' and Q'. (d) Verify that that normal to this second line is in fact $\mathbf{m} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$, confirming that the normal transform really did transform the normal vector to this line properly.

Inline Exercise 10.23: Consider the line ℓ defined by $2x + 3y = 0$ in $\mathbf{R^2}$; it contains the points $P = (3, -2)$ and $Q =$ the origin. (a) What's the normal vector $\mathbf{v}$ to ℓ? (b) If we apply T_3 to all points of ℓ, we get a new line. Compute an equation for $T_3(\ell)$ by finding $T_3(P)$ and $T_3(Q)$, and using the two-point formula. (c) What's the normal vector $\mathbf{n}$ to $T_3(\ell)$?. (d) The matrix $\mathbf{M}_3$ for T_3 is $\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$; $\mathbf{M}_3^{-1}$ is $\begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix}$. Use this to compute the normal transform for T_3, and apply it to $\mathbf{v}$. Your result should be proportional to $\mathbf{n}$.

Inline Exercise 10.24: We assumed that the matrix $\mathbf{M}$ was invertible when we computed the normal transform. Give an intuitive explanation of why if $\mathbf{M}$ is degenerate (i.e., not invertible), it's impossible to define a normal transform. Hint: suppose that $\mathbf{u}$, in the discussion above, is sent to $\mathbf{0}$ by $\mathbf{M}$, but that $\mathbf{u} \cdot \mathbf{n}$ is nonzero.

10.9 More general 3×3 matrices and homogeneous coordinates

We've now unified translation with the other linear transformations through the device of treating the Euclidean plane as the $w = 1$ plane in xyw-space.

Doing so also let us make a clearer model of the distinction between points (things whose xyw-coordinates have $w = 1$) and vectors (things whose xyw-coordinates have $w = 0$). The collection of transformations we now allow — those represented by 3×3 matrices whose last row is $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$ — are called *affine* transformations.

By allowing even more general matrices (i.e., by removing the restriction on the last row) we can extend to the realm of *projective* transformations. You might be asking "Now that you've got all the transformations you were looking for, why do anything more?"

The answer is that when we study perspective, we'll need to use certain transformations that are not affine. The ones we need will exactly correspond to matrices without the last-row restriction. Recall for a moment the operation we did in emulating the perspective-drawing operation shown by Dürer's woodcut: we took xyz-coordinates and computed from them $(x/z, y/z)$. This transformation is by no means linear. The great surprise is that it can be unified with linear transformations in a very natural way, which we'll now show you. But before we do so, let's note something obvious about the Dürer transformation: it's *not defined* when $z = 0$. This phenomenon will show up in our new transformations as well. While affine transformations are defined on all of E^2, projective transformations will in general be defined on all of E^2 except for some line (and when we study E^3, Euclidean 3-space, we'll find projective transformations are undefined on a whole plane, just like the $z = 0$ plane in the Dürer transformation). With this caveat in mind, let's proceed.

Here is the main idea: each point of the $w = 1$ plane can be joined to the origin by a line (see figure 1.18), and almost every line through the origin hits the $w = 1$ plane at some point. (The horizontal lines miss it completely.) Restricting our attention to invertible matrices M, we can now make an important observation. The transformation

$$T_{\mathbf{M}} : \mathbf{R^3} \to \mathbf{R^3} : \mathbf{x} \mapsto \mathbf{Mx}$$

defined by multiplication by the matrix M takes vectors in $\mathbf{R^3}$ to vectors in $\mathbf{R^3}$, but it does something more: it takes lines through the origin to lines through the origin. If ℓ is a line through the origin, then $T(\ell)$ is also a line through the origin. The reason is simple: if $\mathbf{v}$ is a nonzero vector in ℓ, then ℓ consists of all possible multiples $\alpha\mathbf{v}$ of $\mathbf{v}$, where α is a real number. If we compute $T(\alpha\mathbf{v})$, we get $\alpha T(\mathbf{v})$, so T, applied to any point of ℓ, is some multiple of $T(\mathbf{v})$; hence $T(\ell)$ is the line consisting of all multiples of $T(\mathbf{v})$.

Let's also take a closer look at what happens to a small region of E^2 under this transformation, as shown in Figure 1.19. Rays through the origin become horizontal lines; the trapezoid bounded by $x = \pm y$, $x = 1/2$, and $x = 1$ transforms into a rectangle.

Since T can now be seen as sending lines-through-the-origin to lines-through-the-origin, it *almost* defines a transformation of the $w = 1$ plane. Here's the recipe we could follow:

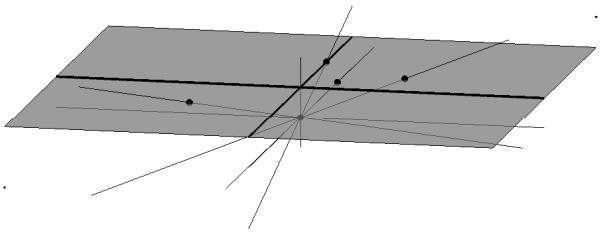

Figure 10.18: Each point of the $w = 1$ plane is the intersection of a line through the origin with the $w = 1$ plane. Each line through the origin, except horizontal ones, meets the $w = 1$ plane in a single point.

fig:homog1

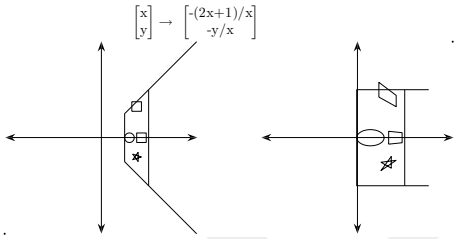

Figure 10.19: The region of E^2 between the lines $y = x$, $y = -x$, $x = 1/2$, and $x = 1$, is transformed to the rectangle defined by $-1 \leq y \leq 1$ and $0 \leq x \leq 1$ square. Things close to $x = 1/2$ get stretched vertically; things closer to $x = 1$ remain unstretched.

fig:perspective-example2

- For a point $P = (x, y) \in E^2$, which we represent by the vector $\mathbf{v} = \begin{bmatrix} x & y & 1 \end{bmatrix}^t$, let ℓ be the line consisting of all multiples of $\mathbf{v}$.
- Compute $T(\ell)$, which is another line, m, in 3-space.
- The line m intersects the $w = 1$ plane at some point $\mathbf{u} = \begin{bmatrix} x', y', 1 \end{bmatrix}^t$, corresponding to a point $P' = (x', y')$.
- Say that the transformation of the $w = 1$ plane sends P to P'.

Of course we don't really need to find *every* point on the line m, just to find the single one that happens to have $w = 1$. We could compute $T(\mathbf{v}) = \begin{bmatrix} u & v & w \end{bmatrix}^t$, and then, since every other point of m is a multiple of this (and vice versa), we could multiply by $1/w$ to get the point $\begin{bmatrix} u/w & v/w & 1 \end{bmatrix}^t$.

The reason this recipe isn't perfect is that it's *possible* that m is a horizontal line, i.e., that the third coordinate of $T(\mathbf{v})$ is zero, so m doesn't intersect the $w = 1$ plane at all. This is an example of the "might be undefined at some points" phenomenon we discussed above. Ignoring this for the moment, let's look at a concrete example: the transformation associated to the matrix

$$\mathbf{M} = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

Our rule says that we should *first* transform the point (x, y), which we represent by $\mathbf{x} = \begin{bmatrix} x & y & 1 \end{bmatrix}^t$, to

$$\mathbf{M}\mathbf{x} = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 2x + 1 \\ y \\ -x \end{bmatrix}$$

and then, because the last coordinate $(-x)$ might not be one, divide through by it, to get

$$\begin{bmatrix} (2x + 1)/(-x) \\ y/(-x) \\ 1 \end{bmatrix} = \begin{bmatrix} (-2x - 1)/x \\ -y/x \\ 1 \end{bmatrix}.$$

This transformation is clearly undefined when $x = 0$, but otherwise it's at least defined. We can see what it does to the house figure in Figure 1.20. Of course, the left side of the house is at $x = 0$, so we cannot transform these points.

With this example in hand, let's look at the process of transforming lines-through-the-origin in 3-space from a different point of view. We regarded the points in the $w = 1$ plane as being in correspondence with lines through the origin (except for a few of them). But we can also regard points on the unit sphere, $\mathbf{S^2}$, as being in correspondence with these lines: each line corresponds to a pair of antipodal (opposite) points; each pair of antipodal points corresponds to a line (see Figure 1.21.

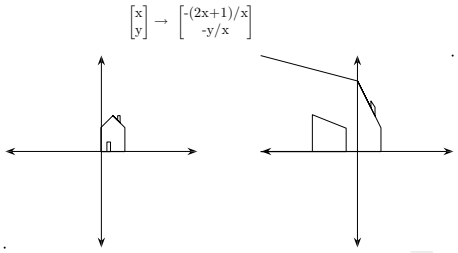

Figure 10.20: The result of the transformation sending the point (x, y) to $(\frac{-2x-1}{x}, \frac{-y}{x})$. Points on the left side of the house, where $x = 0$, are not in the domain, so they cannot be transformed. Other lines that meet $x = 0$, like the bottom of the house and the left side of the roof, are transformed to rays that extend infinitely to the left.

fig:perspective-example

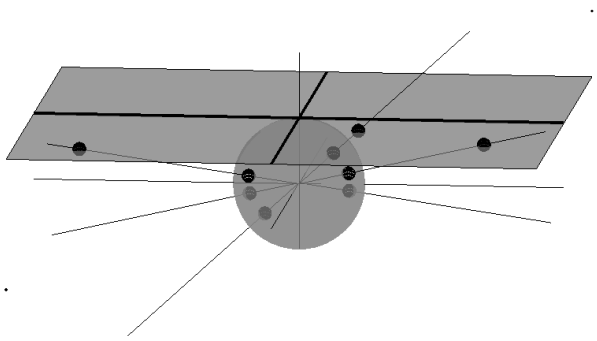

Figure 10.21: Each line through the origin meets the unit sphere, $\mathbf{S^2}$, at a pair of antipodal (opposite) points of the sphere. The line consisting of all multiples of some vector v meets the sphere at the points $\pm\mathbf{v}/\|vvec\|$.

fig:antipodal

Similarly, our linear transformation, defined by $T_{\mathrm{M}}(\mathbf{x}) = \mathbf{Mx}$, sends the points of the sphere to some ellipsoid centered at the origin (this is a consequence of the Singular Value Decomposition theorem). We can take each transformed point and normalize it (divide by its length) to get a point on the sphere. Thus we can see T as a function that takes points on the sphere to points on the sphere, much as we saw it taking points on the $w = 1$ to points on the $w = 1$ plane (mostly).

Each point of the $w = 1$ plane corresponds to a line through the origin, which meets the sphere at some point in the upper hemisphere. The lines *not* corresponding to points in the $w = 1$ plane correspond to points on the equator of the sphere.

Applying our transformation sends points of the sphere to points of the sphere. The ones that get sent to points on the equator corresponds to points that "fall outside the $w = 1$ plane." Thought of in this manner, the $w = 0$ case is not so very alarming: it's just something happening in $\mathbf{R^3}$ that we cannot completely understand by looking at the $w = 1$ plane. By the way, points near the equator of $\mathbf{S^2}$ correspond to nearly-horizontal lines, which correspond to points way out "at the edge" of the $w = 1$ plane. For this reason, points *on* the equator of $\mathbf{S^2}$ are sometimes called *points at infinity*. These are exactly the points whose coordinate triples have the form $\begin{bmatrix} u & v & 0 \end{bmatrix}^t$, with $u^2 + v^2 = 1$. Careful study of projective geometry can lead one to an understanding of why points at infinity and vectors are both represented by triples with zero in the last coordinate [**?**].

There are three last things to note about projective transformations:

- Lines (of the $w = 1$ plane) transform to lines (except where the transformation is not defined).
- Parameterized lines don't transform the way you expect them to.

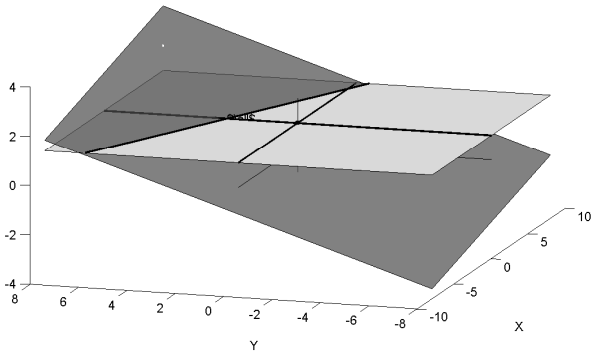

Figure 10.22: The line ℓ in the $w = 1$ plane is the intersection of a plane P through the origin with the $w = 1$ plane. Each point of ℓ corresponds to a line through the origin that lies in P.

fig:projective-line

- Projective transformations transform vectors and covectors in more complicated ways than do affine transformations.

Let's examine these one at a time.

Projective maps transform lines to lines

Consider a line ℓ in the $w = 1$ plane. The points of ℓ correspond to lines through the origin in 3-space, and these lines, taken together, all lie in a plane P (see Figure 1.22). The transformation T_{M}, being linear, takes planes-through-the-origin to planes-through-the-origin. So $T(P)$ is another plane; the intersection of this plane with $w = 1$ is *almost* where T_{M} sends ℓ, when we consider it as a projective transformation (i.e., divide through by the last coordinate). Once again, points at infinity slightly complicate matters. A horizontal vector in P may be transformed to a non-horizontal vector in $T(P)$; this makes a point not in ℓ appear in what we're calling $T(\ell)$. Similarly, a point of ℓ may be transformed by T to a point with last coordinate zero, and hence not correspond to any point of $w = 1$. But the main idea is simple: transforming ℓ, together with its point-at-infinity, gives a new line, together with *its* point-at-infinity.

Transforming parameterized lines

Let's consider a parametric line γ that lies in the $w = 1$ plane, such as

$$\gamma : \mathbf{R} \to \mathbf{R^3} : t \mapsto \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} + t \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}.$$

Suppose we transform each point of this line by $T_{\mathbf{M}}$, where we'll once again use the particular matrix

$$\mathbf{M} = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

We get the line consisting of points of the form

$$\begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} + t \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} t \\ 2+3t \\ 2+t \end{bmatrix}.$$

Dividing through by the third coordinate gives us a line in the $w = 1$ plane consisting of points of the form

$$\begin{bmatrix} \frac{t}{2+t} \\ \frac{2+3t}{2+t} \\ 1 \end{bmatrix}.$$

As written, it's not at all obvious that these points all lie on a line. A bit of algebra, however, let's us rewrite this as

$$\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + \frac{t}{2+t} \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}.$$

These points clearly lie on a line. But what's also clear is that equally-spaced points on the original line (corresponding to $t = 0, 1, 2, \ldots$ for example), are no longer equally spaced on the transformed line (the multipliers for the direction vector, at the corresponding points, are $0, 1/2, 2/3, 3/4, \ldots$). This is a source of many algorithmic bugs: while lines transform to lines under projective transformations, their parameterizations do not transform in an obvious way.

On the other hand, one can still interpolate values after a projective transformation, as we discuss in Section **??**.

Transforming vectors and covectors

Let's make our recipe for transforming points by a projective transformation defined by the matrix M really explicit. We define

$$H : \mathbf{R^3} - \{(x, y, 0) | x, y \in \mathbf{R}\} \to \mathbf{R^3} : (x, y, w) \mapsto (x/w, y/w, 1),$$

and call it the *homogenizing transformation*[4]. Then the projective transformation $S_{\mathbf{M}}$ associated to the matrix $\mathbf{M}$ is defined by

$$S_{\mathbf{M}} : U \to \mathbf{R^3} : \mathbf{x} \mapsto H(T_{\mathbf{M}}(\mathbf{x}))$$

where U is the subset of $\mathbf{R^3}$ consisting of vectors for which $T_{\mathbf{M}}(\mathbf{x})$ does not have zero as its w-coordinate.

For the 3×3 versions of T_1 through T_5, the function $S_{\mathbf{M}}$, restricted to E^2, is the same as $T_{\mathbf{M}}$, because the third coordinate of the result is always one, and so H has no effect.

But as we saw, for the transformation defined by

$$\mathbf{M} = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix},$$

H has an important effect.

One component of this effect is in the way that H transforms vectors. Up until now, all our transformations have had the property that they transform vectors the same way, independent of position (i.e., it made sense to think of vectors as displacements of the entire plane, with no particular "starting point"). Now that we are encountering non-linear, non-affine transformations, we must look at things differently.

The transformation of vectors associated to the matrix transform $\mathbf{x} \mapsto T_{\mathbf{M}}(\mathbf{x}) = \mathbf{M}\mathbf{x}$ is, in fact, the *derivative* of this transformation. And just as when we look at an affine function from $\mathbf{R}$ to $\mathbf{R}$ like $f(x) = 7x + 3$, we find that the derivative $f'(x) = 7$ is a constant, independent of x, the derivative of $T_{\mathbf{M}}$ is also a constant, namely the "vector transformation" that we found earlier, whose matrix form is just $\mathbf{M}$ with its last row changed to zeroes.

The derivative of $S_{\mathbf{M}}$, however, is more complicated: the function $S_{\mathbf{M}}$ is a composition of two others:

$$S_{\mathbf{M}} = H \circ T_{\mathbf{M}}$$

and hence, by the chain rule for derivatives, its derivative is the product of the derivatives of H and $T_{\mathbf{M}}$; in particular, we

$$DS_{\mathbf{M}}(\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}) = DH(T_{\mathbf{M}}(\begin{bmatrix} x \\ y \\ 1 \end{bmatrix})) \cdot DT_{\mathbf{M}}(\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}) \tag{10.24}$$

$$= DH(T_{\mathbf{M}}(\begin{bmatrix} x \\ y \\ 1 \end{bmatrix})) \cdot \mathbf{M} \tag{10.25}$$

[4]This is really a misnomer: the coordinate triples that include a one in the w coordinate are a special case of "homogeneous coordinates," in which two xyw-triples are considered the same if one is a nonzero multiple of the other; *any* nonzero xyz-triple is allowed. In this sense, the coordinates are homogeneous – there's no special preference given to x and y over w. Our map H should really be called a *dehomogenizing transform,* but the other name is already standard.

What is DH? Writing out H as

$$H(\begin{bmatrix} x \\ y \\ w \end{bmatrix}) = \begin{bmatrix} x/w \\ y/w \\ 1 \end{bmatrix},$$

we compute its derivative (Jacobian matrix):

$$DH(\begin{bmatrix} x \\ y \\ w \end{bmatrix}) = \begin{bmatrix} 1/w & 0 & -x/w^2 \\ 0 & 1/w & -y/w^2 \\ 0 & 0 & 0 \end{bmatrix}.$$

That matrix clearly depends on x, y and w, hence DS is not a constant: the way that vectors get transformed depends on where they start. As an example, return to the case of

$$\mathbf{M} = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

Compute for yourself $S_{\mathbf{M}}$ applied to the points $P_1 = (1, 0)$ and $P_2 = (1, 1)$, which differ by the vector $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$. What's the vector between the results? Compare this to the vector between the transforms of the points $Q_1 = (2, 0)$ and $Q_2 = (2, 1)$. Both pairs differ by $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ in the domain, but the differences in the codomain are not at all the same.

A very similar thing happens with covectors transformed by projective transformations: the transformation varies from point to point.

The details here are not particularly important, because the principal use of transformed vectors is in simplifying certain computations. For example, one might hope that computing $\mathbf{v}\cdot\mathbf{w}$ could be done before *or* after performing a rotation (which is correct). If $\mathbf{v}$ is the normal vector to a surface, and $\mathbf{w}$ is the incoming light vector, this would mean that one could compute the amount that the light hits the surface either before or after rotation. If the rotation happened to turn $\mathbf{w}$ into, say, $\mathbf{e}_3$, then the dot-product *after* rotation is particularly easy to compute; if one was going to rotate things anyhow, you'd rather do the after-transform dot-product than the before-transform dot-product. The one place where we'll see projective transformations really used in graphics will be in the "camera de-perspectivizing transformation,", the 3D analog of the one shown in Figure 1.19, in which objects in the field of view of a pinhole camera are transformed to be in the field of view of a "parallel camera"; the lesson to be remembered is that anything we need to compute about the transformed objects should be computed *before* the transformation.

10.9.1 Transformations versus interpolation

When you rotate a book on your desk by thirty degrees counterclockwise, as you do so, the book is rotated by each intermediate amount between zero and thirty degrees. But when we "rotate" the house in our diagram by thirty degrees, we simply compute the final position of each point of the house. In no sense has it passed through any intermediate positions. In the more extreme case of rotation by 180 degrees, the resulting transformation is exactly the same as the "uniform scale by -1" transformation. And in the case of rotation by 360 degrees, the resulting transformation is the identity.

This reflects a *limitation in modeling*. The use of matrix transformations to model transformations of ordinary objects captures only the relationship between initial and final positions, and not the means by which the object got from the initial to the final position.

Much of the time, this distinction is unimportant: we want to put an object into a particular pose, so we apply some sequence of transformations to it (or its parts). But sometimes it can be quite significant: we might instead want to show the object being transformed from its initial state to its final state. An easy, but rarely useful, approach is to linearly interpolate each point of the object from its initial to its final position. If we do this for the "rotation by 180 degrees" example, at the halfway point the entire object is collapsed to a single point; if we do it from the "rotation by 360 degrees" example, the object never appears to move at all! The problem is that what we *really* want is to find interpolated versions of our *descriptions* of the transformation rather than of the transformations themselves. (Thus, to go from the initial state to "rotated by 360", we'd apply "rotate by s" to the initial state, for each value of s from 0 to 360.)

But sometimes students confuse a transformation like "multiplication by the identity matrix," with the way it was specified "rotate by 360 degrees," and can be frustrated with the impossibility of "dividing by two" to get a rotation by 180 degrees, for instance. This is particularly annoying when one has access only to the matrix form of the transformation, rather than the initial specification; in that case, as the examples show, there's no hope for a general solution to the problem of "doing a transformation partway." On the other hand, there *is* a solution that often gives reasonable results in practice, especially for the problem of interpolating two rather similar transformations (e.g., interpolating between rotating by 20 degrees and rotation by 30 degrees), which often arises. We'll discuss this in Chapter 11. In preparation for that, however, we'll briefly discuss rotations of the plane and their relation to the unit circle and the exponential map.

10.10 The unit circle, rotations of the plane, and the exponential map

This section involves somewhat more difficult mathematics than the previous ones, and may be skipped on a first reading.

The set of rotations of $\mathbf{R^2}$ consists of matrices like the one for T_1, as we've said before, but with 30° replaced by all possible angles, i.e., matrices of the form

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{10.26}$$

which rotates $\mathbf{e}_1$ to the vector $\begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$.

This set of matrices is called $\mathbf{SO}(2)$, the *special orthogonal group in two dimensions*. These matrices are said to be "orthogonal" because their columns are unit vectors that are orthogonal[5] to each other; such a matrix always has determinant ± 1; among orthogonal matrices, there are those whose columns form a positively-oriented coordinate system; these are termed "special," and are exactly those with determinant $+1$. When we look at $\mathbf{SO}(3)$, exactly analogous properties will hold.

When we encounter new sets (like $\mathbf{SO}(2)$) we usually try to *parameterize* them, i.e., we try to find a mapping from some familiar space (like Euclidean space) to the set, a mapping that's bijective (if possible) or "nearly" bijective if it's not. The map-maker's problem is an instance of this: the map-maker tries to make a comprehensible picture of the surface of the whole earth on a sheet of paper, i.e., the map-maker seeks a function from a rectangle (the piece of paper) to the sphere. It's impossible to do this in a continuous and bijective way, but by deciding to have the international dateline appear both at the left and right sides of the map, and to have all points along the top edge correspond to the north pole, and all points on the bottom edge correspond to the south pole, the map-maker creates a map that is bijective except at the poles and the international dateline. Such maps (called Mercator projections) do distort sizes, but they help us understand something about the sphere in terms of the familiar flat plane. In the same way, we describe points of the circle by naming them with angles, thus parameterizing the circle by the real line. It's true that multiple angles (e.g., 0, 2π, 4π, ...) all correspond to the same point of the circle, but being able to speak of points in the circle in terms of real numbers is very convenient.

We'll now proceed to examine $\mathbf{SO}(2)$ this way — through a parameterization — in considerable detail, even though $\mathbf{SO}(2)$ is topologically just a circle, hence not very difficult to understand. The reason for this study is to

[5] Because the columns are both orthogonal *and* normalized (i.e., unit-length), it would make sense to call such matrices orthonormal; unfortunately, the terms "orthogonal" is well-established.

provide intuition for the study of $\mathbf{SO}(3)$ (the set of 3×3 rotation matrices), which will follow an exactly parallel path; because $\mathbf{SO}(3)$ is more complicated, it's best to learn the key ideas in a case that's easy to visualize.

The function R of Equation 1.26 establishes a one-to-one correspondence between points of the circle (indicated by the angle, θ) and rotation matrices. The inverse function is given by

$$R^{-1} : \mathbf{SO}(2) \to \mathbf{R} : \begin{bmatrix} a & -b \\ b & a \end{bmatrix} \mapsto \text{atan2}(b, a),$$

which returns an angle between $-\pi$ and π, corresponding to a point on the unit circle.

We'll now look at this correspondence a little differently: any 2×2 matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ can be thought of as corresponding to an element of $\mathbf{R^4}$, namely $\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$. Under this correspondence, $\mathbf{SO}(2)$ becomes a set of points in $\mathbf{R^4}$, namely those of the form $\begin{bmatrix} \cos\theta \\ -\sin\theta \\ \sin\theta \\ \cos\theta \end{bmatrix}$. We'll denote this set $\mathbf{SO}(2)_4$. These points all lie in a two-dimensional plane in $\mathbf{R^4}$, however, defined (in $xyzw$-coordinates) by $x = w$ and $y = -z$. While it's difficult to "see" things in 4-space, we can certainly look at a two-dimensional subspace of 4-space. The vectors $\mathbf{u}_4 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ and $\mathbf{v}_4 = \begin{bmatrix} 0 \\ -1 \\ 1 \\ 0 \end{bmatrix}$ form a coordinate system[6] for the plane containing $\mathbf{SO}(2)_4$; in particular, we can write

$$\begin{bmatrix} \cos\theta \\ -\sin\theta \\ \sin\theta \\ \cos\theta \end{bmatrix} = \cos\theta\mathbf{u}_4 + \sin\theta\mathbf{v}_4, \tag{10.27}$$

showing that considered as a subset of $\mathbf{R^4}$, $\mathbf{SO}(2)_4$ is actually circular – it's a perfect circle in the plane spanned by $\mathbf{u}_4$ and $\mathbf{v}_4$, as shown in Figure 1.23

Inline Exercise 10.25: $\mathbf{SO}(2)_4$ is a perfect circle; what is its radius?

When we study $\mathbf{SO}(3)$ later, we'll see that it, too, can be examined this way. It can be considered as lying in $\mathbf{R}^9$; unfortunately, it turns out *not* to be just a sphere, but somewhat more complicated.

[6] Everything associated with the placing of $\mathbf{SO}(2)$ in $\mathbf{R^4}$ will be decorated with the subscript 4.

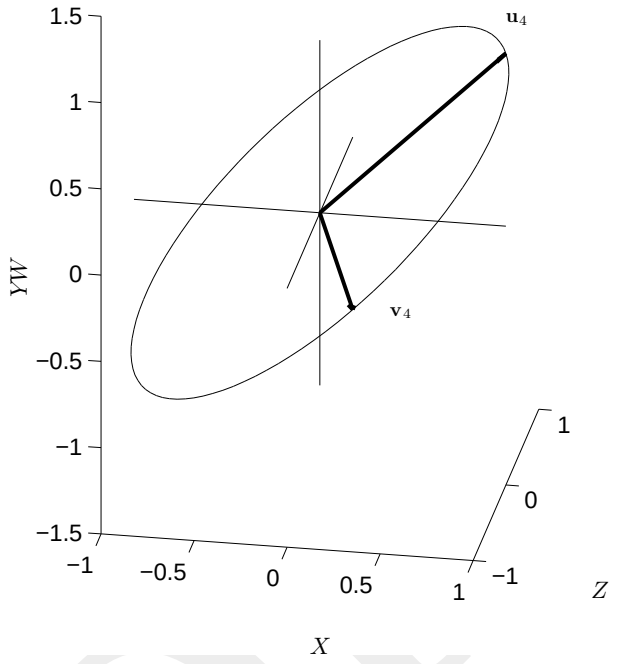

Figure 10.23: The set of all 2×2 rotation matrices, viewed as a subset of $\mathbf{R^4}$. We've projected $\mathbf{R^4}$ to $\mathbf{R^3}$ so that the x- and z-axes are preserved, but the y- and w-axes become coincident, and the resulting direction is given the label yw. The projections of the vectors $\mathbf{u}_4$ and $\mathbf{v}_4$ are shown as well (green and red, respectively); they are perpendicular in $\mathbf{R^4}$, but their projections into $\mathbf{R^3}$ are not.

fig:so2-in-r4

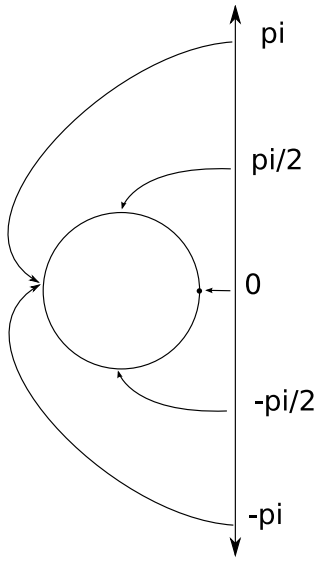

Figure 10.24: The mapping $t \mapsto (\cos t, \sin t)$ takes points of the real line to points of the circle; we've drawn the real line vertically to the right of the circle to show why this is sometimes called the "wrapping function." Notice that the point 0 on the real line corresponds to the point $(1, 0)$ of the circle.

fig:wrapping-function

To return to $\mathbf{SO}(2)$: we now can see $\mathbf{SO}(2)$ as a circle sitting in a plane within $\mathbf{R^4}$. We'll therefore briefly discuss circles, and then apply our discussion to $\mathbf{SO}(2)_4$.

It's conventional to parameterize a circle by a line, with the usual parameterization being

$$\gamma : \mathbf{R} \to \mathbf{R^2} : t \mapsto (\cos t, \sin t),$$

which is sometimes described by saying that γ "takes the real line and wraps it around the circle," as shown in figure 1.24. This easy-to-understand function is actually an instance of something far more general, called an *exponential function*. Although the function involves only sines and cosines in its current form, we'll see that it's closely related to a function defined by exponentiation, which is where the name arises.

Figure 1.24 shows the domain of the wrapping function as a line to the right of the circle. We'll now alter things slightly, and consider essentially the same function, but with the tangent line to the circle at $(1, 0)$ as its domain. This tangent line T can be thought of as consisting of all vectors of the form

$$t \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

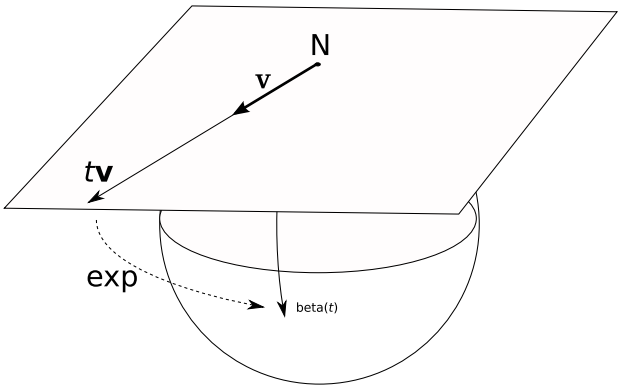

Figure 10.25: A ray in the tangent plane to the sphere at the north pole can be "wrapped" onto the sphere so that it traverses a great circle. We call this mapping from the tangent plane to the sphere "exp". If the ray is parameterized as $N + t\mathbf{v}$, and we define $\beta(t) = \exp(N + t\mathbf{v})$, then $\beta'(0) = \mathbf{v}$, i.e., the velocity vector of the wrapped curve at time zero is the same as the velocity vector of the ray at time zero.

fig:exponential-sphere

drawn so that they originate from $(1, 0)$; with that understanding our function now looks like this:

$$\hat{\gamma} : T \to \mathbf{R^2} : t \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mapsto (\cos t, \sin t).$$

This may not seems like a significant change, but we can view this function as an instance of a more general class of functions: take any smooth shape (like a sphere, for instance), and a point of that shape (like the north pole, $N = (0, 1, 0)$); the tangent-space to the shape at that point (a horizontal plane, in this example) can now be "wrapped" onto the shape in much the same way (see Figure 1.25).

In the case of the sphere, it's not even very difficult to write down the exponential map explicitly: any vector in the tangent plane to the north pole can be written in the form $t\mathbf{v}$, where $\mathbf{v}$ is a *unit* vector that lies in the xz-plane. The vector $\mathbf{0}$ is sent, by exp, to the point

$$\exp(t\mathbf{v}) = \cos(t)N + \sin(t)\mathbf{v}.$$

Since $\mathbf{v}$ is a horizontal vector, it can be written $\begin{bmatrix} x \\ 0 \\ z \end{bmatrix}$, so this becomes

$$\exp(t\mathbf{v}) = \begin{bmatrix} x \sin t \\ \cos t \\ z \sin t \end{bmatrix}.$$

Inline Exercise 10.26: The vector $\mathbf{w} = \pi \begin{bmatrix}1\\0\\0\end{bmatrix}$ is sent, by $\exp$, to the south pole. Find another vector that's sent there.

There are many different maps from the tangent plane at the north pole onto the sphere, but the one we've defined has some very nice properties:

- straight lines through the north pole are sent to great circles, which are paths of shortest distance[7], and
- a parameterized ray $t \mapsto t\mathbf{v}$ through the north pole exponentiates to a parameterized curve through the north pole whose velocity, as it passes the north pole, is exactly $\mathbf{v}$.

Returning briefly to the one-dimensional case, we see that analogues of these properties hold in that situation: the wrapping function takes the tangent line at $(1,0)$ and wraps it onto the circle (which *is* a great circle, in the sense of having the shortest-path property); when we exponentiate $t \mapsto t\begin{bmatrix}0\\1\end{bmatrix}$, we get $t \mapsto (\cos t, \sin t)$, whose velocity, at $t=0$, is exactly $\begin{bmatrix}0\\1\end{bmatrix}$,

Now let's move to the larger context: we have $\mathbf{SO}(2)_4$ sitting in $\mathbf{R^4}$ as a circle in the $x = w, y = -z$ plane. The identity matrix corresponds to the point $I_4 = \begin{bmatrix}1\\0\\0\\1\end{bmatrix}$ of the circle; the direction of the tangent line to this circle at this point (which we can find by differentiating the expression in Equation 1.10 with respect to θ at $\theta = 0$) is simply $\mathbf{v}_4 = \begin{bmatrix}0\\-1\\1\\0\end{bmatrix}$, which corresponds to the matrix

$$\begin{bmatrix}0 & -1\\1 & 0\end{bmatrix}.$$

The exponential map from the tangent line to $\mathbf{SO}(2)_4$ then can be written quite simply: the point $t\mathbf{v}_4$ of the tangent line is sent to the point $\cos(t)\mathbf{u}_4 + \sin(t)\mathbf{v}_4$. In explicit coordinates, this means that

$$\exp\begin{bmatrix}0\\-t\\t\\0\end{bmatrix} = \begin{bmatrix}\cos t\\-\sin t\\\sin t\\\cos t\end{bmatrix}$$

[7] This is to say: for any two points on a great circle that are close enough (in the case of the sphere, "close enough" means "not opposite each other"), the shortest path between the points is actually an arc of the great circle. Paths on more general objects with this property are called "geodesics."

Back in matrix form, this would say that

$$\exp\begin{bmatrix}0 & -t\\ t & 0\end{bmatrix} = \begin{bmatrix}\cos t & -\sin t\\ \sin t & \cos t\end{bmatrix}.$$

This is where things get surprising. It turns out that if we take the Taylor series for $\exp$ as a function from $\mathbf{R}$ to $\mathbf{R}$, namely

$$\exp(x) = 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \ldots,$$

and apply it willy-nilly to a square matrix (replacing the first 1 with the identity matrix), the infinite series still converges (the proof of convergence is very similar to the one for the Taylor series). In fact, the resulting function from square matrices to square matrices (of a given size) has many of the properties of the exponential map for real numbers, things like

- $\exp(\mathbf{0}) = \mathbf{I}$.
- $\exp(k\mathbf{M}) = \exp(\mathbf{M})^k$ for $k = \ldots, -2, -1, 0, 1, 2, \ldots$.
- $\exp(\mathbf{A} + \mathbf{B}) = \exp(\mathbf{A})\exp(\mathbf{B})$ (but *only* if $\mathbf{AB} = \mathbf{BA}$).
- $\exp(-\mathbf{M}) \cdot \exp(\mathbf{M}) = \exp(\mathbf{0}) = \mathbf{I}$, so $\exp(-\mathbf{M})$ is the *inverse* of $\exp(\mathbf{M})$.
- $\exp$ is a smooth function, and its derivative at 0 is the identity.

There are some other properties that only make sense for matrices, like

- $\exp(\mathbf{M}^t) = \exp(\mathbf{M})^t$.
- $det\exp(\mathbf{M})) = \exp(tr(\mathbf{M}))$, where $tr(\mathbf{M})$ is the sum of the diagonal entries of $\mathbf{M}$.

These properties, taken together, tell us that $\exp$ takes *any* matrix $\mathbf{A}$ and sends it to a matrix $\exp(\mathbf{A})$ that is invertible (the inverse is just $\exp(-\mathbf{A})$, according to the third property). If we let M_n denote the set of all $n \times n$ matrices, and $GL(n)$ denote the invertible ones, then we see that

$$\exp : M_n \to GL(n).$$

(In the case of the exponential function on real numbers, we'd say that $\exp$ takes $\mathbf{R}$ to $\mathbf{R}^+$, where we're making the natural identification between the number x and the matrix $\begin{bmatrix}x\end{bmatrix}$).

Let's once again make this concrete with an example: letting $\mathbf{J}$ denote

$$\mathbf{J} = \begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix},$$

which is a representative element of the tangent line to $\mathbf{SO}(2)$ at the identity element, let's compute $\exp(\mathbf{J})$.

Inline Exercise 10.27: Show that $\mathbf{J}^2 = -\mathbf{I}$, $\mathbf{J}^3 = -\mathbf{J}$, and $\mathbf{J}^4 = \mathbf{I}$. Give a general form for $\mathbf{J}^n$, where n is a positive integer.

By definition,

$$\exp(\mathbf{J}) = \mathbf{I} + \frac{1}{1!}\mathbf{J} + \frac{1}{2!}\mathbf{J}^2 + \frac{1}{3!}\mathbf{J}^3 + \frac{1}{4!}\mathbf{J}^4 + \frac{1}{5!}\mathbf{J}^5 + \dots \tag{10.28}$$

$$= \mathbf{I} + \frac{1}{1!}\mathbf{J} + \frac{1}{2!}(-\mathbf{I}) + \frac{1}{3!}(-\mathbf{J}) + \frac{1}{4!}(-\mathbf{I}) + \frac{1}{5!}\mathbf{J} + \dots \tag{10.29}$$

$$= \mathbf{I} + \frac{1}{2!}(-\mathbf{I}) + \frac{1}{4!}(-\mathbf{I}) + \dots \frac{1}{1!}\mathbf{J} + \frac{1}{3!}(-\mathbf{J}) + \frac{1}{5!}\mathbf{J} + \dots \tag{10.30}$$

$$\tag{10.31}$$

where we have simply gathered the even and odd terms[8] Further simplifying, we get

$$\exp(\mathbf{J}) = \mathbf{I}(1 - \frac{1}{2!} + \frac{1}{4!} - \dots) + \mathbf{J}(\frac{1}{1!} - \frac{1}{3!} + \frac{1}{5!} - \dots) \tag{10.32}$$

$$= \mathbf{I}\cos(1) + \mathbf{J}\sin(1), \tag{10.33}$$

where the last simplification uses the Taylor series for sine and cosine. An essentially identical computation shows that

$$\exp(t\mathbf{J}) = \mathbf{I}\cos(t) + \mathbf{J}\sin(t) \tag{10.34}$$

$$= \begin{bmatrix} \cos t & -\sin t \\ \sin t & \cos t \end{bmatrix}. \tag{10.35}$$

So the mapping from the tangent line to $\mathbf{SO}(2)$ at the identity (which consists of all multiples of the matrix $\mathbf{J}$) to $\mathbf{SO}(2)$ turns out to be *exactly* the matrix exponential, thus justifying the use of the name $\exp$.

The method just shown for computing $\exp$ on a matrix — explicit multiplication and simplification — is really practical only in cases where the matrix has special properties like those of $\mathbf{J}$. In general, one computes the exponential by numerical methods; because of the $n!$ in the denominator of the nth term, the series converges rapidly.

The computation of $\exp(\mathbf{J})$ demonstrates a more general property: the matrix $\mathbf{J}$ is *skew-symmetric*, i.e., $\mathbf{J}^t = -\mathbf{J}$, and $\exp(\mathbf{J})$ turned out to be a rotation. In general, $\exp$ takes skew-symmetric matrices to rotations, even in higher dimensions. We'll use this again when we study $\mathbf{SO}(3)$.

The last property of $\exp$ — that its derivative at $\mathbf{0}$ is $\mathbf{I}$ — tells us that for $\mathbf{M}$ a small matrix (all entries near zero), we have

$$\exp(\mathbf{0} + \mathbf{M}) \approx \exp(\mathbf{0}) + \mathbf{I}\mathbf{M} = \mathbf{I} + \mathbf{M};$$

when we apply this to $\mathbf{M} = t\mathbf{J}$, for small t, we find that

$$\exp(t\mathbf{J}) \approx \mathbf{I} + t\mathbf{J} = \begin{bmatrix} 1 & -t \\ t & 1 \end{bmatrix},$$

[8] The series turns out to be absolutely convergent, which justifies this.

which can frequently be a useful approximation or a small rotation.

Stepping back from our computations, we see several things:

- Exponential takes skew-symmetric matrices to rotations.
- The set of skew-symmetric matrices is very simple: it's just all multiples of **J**, i.e., it's a line, while the set of rotations is slightly more complicated, corresponding, as it does, to a circle in a plane in 4-space.
- The exponential map is one-to-one if we limit our attention to small enough skew-symmetric matrices, but for larger rotations (like $\pm\pi\mathbf{J}$), two different matrices can exponentiate to the same rotation.

All of these characteristics will reappear in the case of 3×3 rotation matrices in chapter 11. It's often helpful to understand those by looking back at the 2×2 case discussed here.

10.11 Further reading

The representation of translation, rotation, scaling, shearing, and even projective transformations via matrices is widely discussed in mathematics texts on projective geometry [**?**]. The particulars of what happens when one views such transformations by making (almost) all homogeneous coordinate triples have the last coordinate be one is not so much discussed, however.

The division by the last coordinate can produce unexpected results when $w < 0$. One can also consider a form of geometry in which two triples are considered the same if one is a *positive* multiple of the other, rather than any nonzero multiple. While this is more or less Riemann's *spherical geometry,* it's best explained for graphics by Stolfi [**?**], who called it "oriented projective geometry."

The "normal transform" that we've described is the inverse of what mathematicians refer to as the *adjoint transformation*. If T is a transformation from a vector space V to a vector space W, then the adjoint of T is a transformation from the dual-space of W to the dual-space of V. In graphics, however, we usually want to go the other direction, and thus invert that adjoint. While the inverse only exists if the original transformation is nondegenerate, the mathematicians' adjoint is always defined. Advanced linear algebra texts like Hoffman and Kunze [**?**] discuss this.

The relation between matrix exponentials and rotations, and the generalization to higher dimensions, is central to the study of *Lie Groups*, which are much used in robotics as well. Adams [**?**] provides a fine mathematical introduction.

Exercise 10.6: (a) [Not too hard] Use the SVD to conclude that any linear transformation T from $\mathbf{R^2}$ to $\mathbf{R^2}$ sends some pair $\mathbf{v}_1$, $\mathbf{v}_2$ of orthogonal unit vectors to another pair $\mathbf{w}_1$, $\mathbf{w}_2$ of vectors with $\mathbf{w}_1 \cdot \mathbf{w}_2 = 0$. Conclude that there's some rectangle in $\mathbf{R^2}$ that's sent by T to another rectangle in $\mathbf{R^2}$ (although the second rectangle might have two sides of length zero). (b) [More difficult, for the mathematically inclined]. Here's a way to prove the SVD exists for a 2×2 matrix $\mathbf{M}$. Let $T(\mathbf{x}) = \mathbf{M}\mathbf{x}$. If T is degenerate (like example T_5), then there's a unit vector $\mathbf{v}$ with $T(\mathbf{v}) = \mathbf{0}$; explain why $\mathbf{v}$ and $\mathbf{v}^\perp$ form the edges of a rectangle that's sent to another rectangle. On the other hand, if T is not degenerate, consider a pair of unit vectors $\mathbf{v}_1(t), \mathbf{v}_2(t)$ in $\mathbf{R^2}$, the first at angle t to the x-axis, the second at angle $t + \pi/2$. Let $\mathbf{w}_1(t) = T(\mathbf{v}_2(t))$, and $\mathbf{w}_2(t) = T(\mathbf{v}_2(t))$. Let $\theta(t)$ be the angle between $\mathbf{w}_1(t)$ and $\mathbf{w}_2(t)$. (i) Explain why θ is well-defined. (ii) Explain why $\theta(\pi/2) = \pi - \theta(0)$. (iii) If $\theta(0) = \pi/2$, then we're done, as $\mathbf{v}_1(t)$ and $\mathbf{v}_2(t)$ are the edge-vectors of the desired rectangle. If not, suppose that $\theta(0) < \pi/2$. Explain why there's some $t_\star$ between 0 and $\pi/2$ with $\theta(t) = \pi/2$. (iv) Finally, letting $\mathbf{u}_i$ be $\mathbf{w}_i(t_\star)/\|\mathbf{w}_i(t_\star)\|$ for $i = 1, 2$, explain why $\mathbf{M}$ can be written in the form

$$\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{V}^t,$$

where $\mathbf{U}$ has $\mathbf{u}_i (i = 1, 2)$ as columns, $\mathbf{V}$ has $\mathbf{v}_i (i = 1, 2)$ as columns, and $\mathbf{D}$ is diagonal.