L6LinearTransformations

February 19, 2015

1 Linear Transformations

So far we've been treating the matrix equation

$$A\mathbf{x} = \mathbf{b}$$

as simply another way of writing the vector equation

$$x_1\mathbf{a_1} + \cdots + x_n\mathbf{a_n} = \mathbf{b}.$$

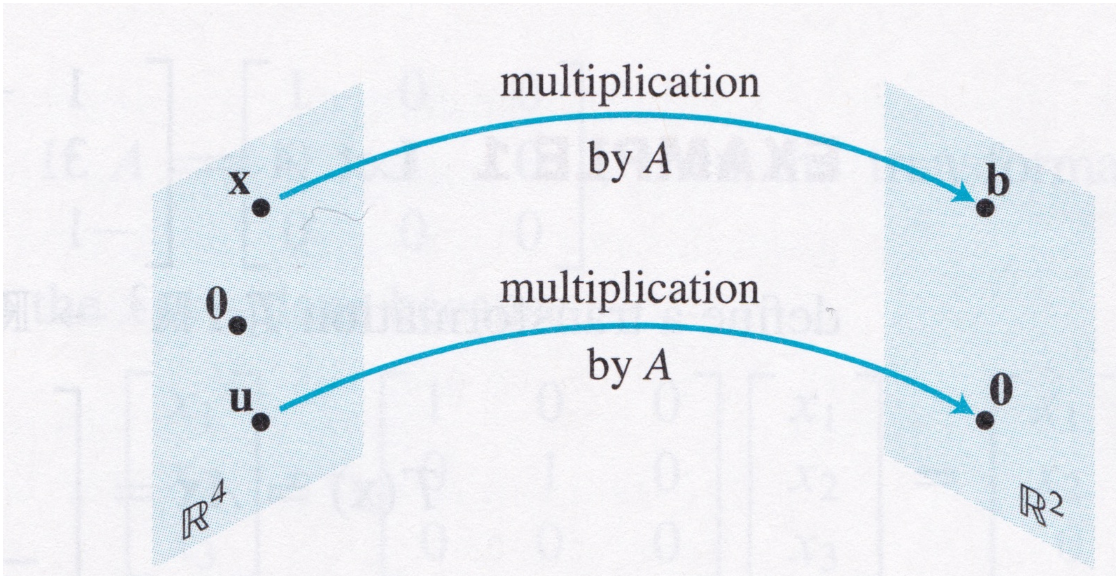
However, we'll now think of the matrix equation in a new way: we will think of A as "acting on" the vector $\mathbf{x}$ to form a new vector $\mathbf{b}$.

For example, let's let $A = \begin{bmatrix} 4 & -3 & 1 & 3 \\ 2 & 0 & 5 & 1 \end{bmatrix}$. Then we find:

$$A\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix} \quad \text{and} \quad A\begin{bmatrix} 1 \\ 4 \\ -1 \\ 3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

In other words, if $\mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$, then A *transforms* $\mathbf{x}$ into $\mathbf{b}$.

Likewise, if $\mathbf{u} = \begin{bmatrix} 1 \\ 4 \\ -1 \\ 3 \end{bmatrix}$, then A transforms $\mathbf{u}$ into the $\mathbf{0}$ vector.



This gives a **new** way of thinking about solving $A\mathbf{x} = \mathbf{b}$. We are "searching" for the vectors $\mathbf{x}$ in $\mathbb{R}^4$ that are transformed into $\mathbf{b}$ in $\mathbb{R}^2$ under the "action" of A.

So the mapping from $\mathbf{x}$ to $A\mathbf{x}$ is a **function** from one set of vectors (those in $\mathbb{R}^4$) to another. We have moved out of the familiar world of functions of one variable: we are now thinking about functions that transform a vector into a vector. Or, put another way, transform multiple variables into multiple variables.

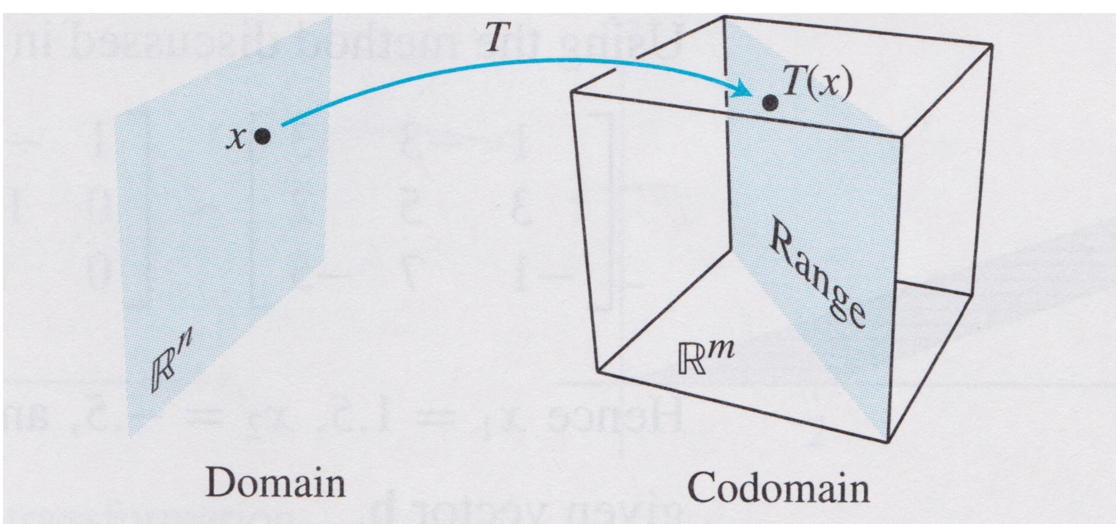
Some terminology:

A **transformation** (or **function** or **mapping**) T from $\mathbb{R}^n$ to $\mathbb{R}^m$ is a rule that assigns to each vector $\mathbf{x}$ in $\mathbb{R}^n$ a vector $T(\mathbf{x})$ in $\mathbb{R}^m$. The set $\mathbb{R}^n$ is called the **domain** of T, and $\mathbb{R}^m$ is called the **codomain** of T. The notation:

$$T : \mathbb{R}^n \to \mathbb{R}^m$$

indicates that the domain of T is $\mathbb{R}^n$ and the codomain is $\mathbb{R}^m$.

For $\mathbf{x}$ in $\mathbb{R}^n$, the vector $T(\mathbf{x})$ is called the **image** of $\mathbf{x}$ (under T). The set of all images $T(\mathbf{x})$ is called the **range** of T.



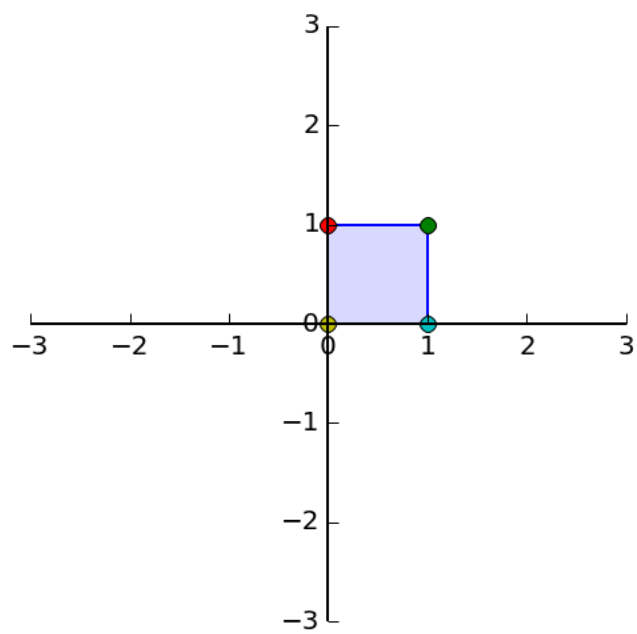
Let's do an example. Let's say I have these points in $\mathbb{R}^2$:

$$\left[\begin{array}{c}0\\1\end{array}\right], \left[\begin{array}{c}1\\1\end{array}\right], \left[\begin{array}{c}1\\0\end{array}\right], \left[\begin{array}{c}0\\0\end{array}\right]$$

Where are these points located?

```
In [5]: square = np.array([[0.0,1,1,0],[1,1,0,0]])
        ax = dm.plotSetup(-3,3,-3,3)
        print square
        dm.plotSquare(square)
```

```
[[ 0.  1.  1.  0.]
 [ 1.  1.  0.  0.]]
```



Now let's transform each of these points according to the following rule. Let

$$A = \left[\begin{array}{cc}1 & 1.5\\0 & 1\end{array}\right].$$

We define $T(\mathbf{x}) = A\mathbf{x}$. Then we have

$$T : \mathbb{R}^2 \to \mathbb{R}^2.$$

What is the image of each of these points under T?

$$A\left[\begin{array}{c}0\\1\end{array}\right] = \left[\begin{array}{r}1.5\\1\end{array}\right]$$

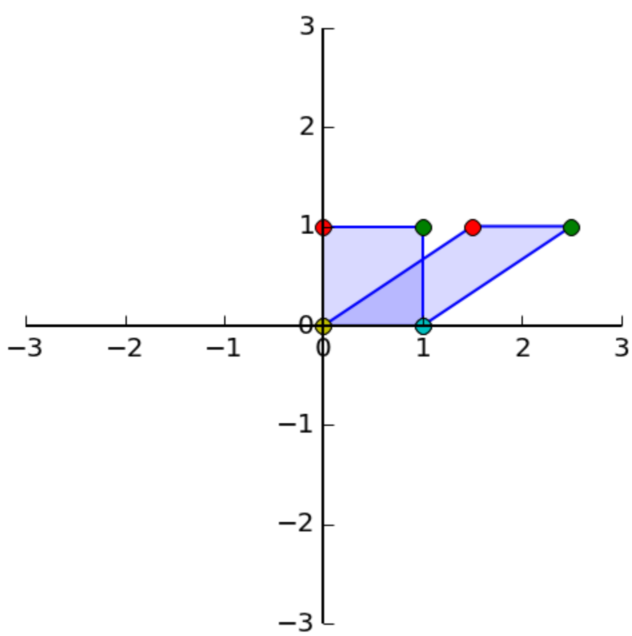
$$A\left[\begin{array}{c}1\\1\end{array}\right] = \left[\begin{array}{r}2.5\\1\end{array}\right]$$

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

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

In [6]:
```
ax = dm.plotSetup(-3,3,-3,3)
print "square = "
print square
dm.plotSquare(square)
#
# create the shear matrix
shear = np.array([[1.0, 1.5],[0.0,1.0]])
print "shear matrix = "
print shear
#
# apply the shear matrix to the square
ssquare = np.zeros(np.shape(square))
for i in range(4):
    ssquare[:,i] = dm.AxVS(shear,square[:,i])
print "sheared square = "
print ssquare
dm.plotSquare(ssquare)
```

```
square =
[[ 0.  1.  1.  0.]
 [ 1.  1.  0.  0.]]
shear matrix =
[[ 1.   1.5]
 [ 0.   1. ]]
sheared square =
[[ 1.5  2.5  1.   0. ]
 [ 1.   1.   0.   0. ]]
```



This sort of transformation, where points are successively slid sideways, is called a **shear** transformation.

1.1 Linear Transformations

By the properties of matrix-vector multiplication, we know that the transformation $\mathbf{x} \mapsto A\mathbf{x}$ has the properties that

$$A(\mathbf{u} + \mathbf{v}) = A\mathbf{u} + A\mathbf{v} \quad \text{and} \quad A(c\mathbf{u}) = cA\mathbf{u}$$

for all $\mathbf{u}, \mathbf{v}$ in $\mathbb{R}^n$ and all scalars c.

We are now ready to define one of the most fundamental concepts in the course: the concept of a *linear transformation.*

(You are now finding out why the subject is called linear algebra!)

Definition. A transformation T is **linear** if: 1. $T(\mathbf{u} + \mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v})$ for all $\mathbf{u}, \mathbf{v}$ in the domain of T; and 2. $T(c\mathbf{u}) = cT(\mathbf{u})$ for all scalars c and all $\mathbf{u}$ in the domain of T.

To fully grasp the significance of what a linear transformation is, don't think of just matrix-vector multiplication. Think of T as a function in more general terms.

The definition above captures a *lot* of functions that are not matrix-vector multiplication. For example, think of:

$$T(x) = \int_0^1 x(t)\,dt$$

Is T a linear function?

1.2 The Matrix of a Linear Transformation

Not all linear transformations are matrix-vector multiplications. But, **every linear transformation from vectors to vectors is a matrix multiplication.**

Theorem. Let $T : \mathbb{R}^n \to \mathbb{R}^m$ be a linear transformation. There there is a unique matrix A such that

$$T(\mathbf{x}) = A\mathbf{x} \quad \text{for all } \mathbf{x} \in \mathbb{R}^n.$$

In fact, A is the $m \times n$ matrix whose jth column is the vector $T(\mathbf{e_j})$, where $\mathbf{e_j}$ is the $j$the column of the identity matrix in $\mathbb{R}^n$:

$$A = [T(\mathbf{e_1}) \ldots T(\mathbf{e_n})].$$

Proof. Write

$$\mathbf{x} = I\mathbf{x} = [\mathbf{e_1} \ldots \mathbf{e_n}]\,\mathbf{x}$$

$$= x_1\mathbf{e_1} + \cdots + x_n\mathbf{e_n}.$$

Because T is linear, we have:

$$T(\mathbf{x}) = T(x_1\mathbf{e_1} + \cdots + x_n\mathbf{e_n})$$

$$= x_1T(\mathbf{e_1}) + \cdots + x_nT(\mathbf{e_n})$$

$$= [T(\mathbf{e_1}) \ldots T(\mathbf{e_n})] \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = A\mathbf{x}.$$

This is a hugely powerful tool. Let's say we start from a linear transformation; we can use this to find the matrix that implments that linear transformation.

For example, let's see how to compute the linear transformation that is a rotation.

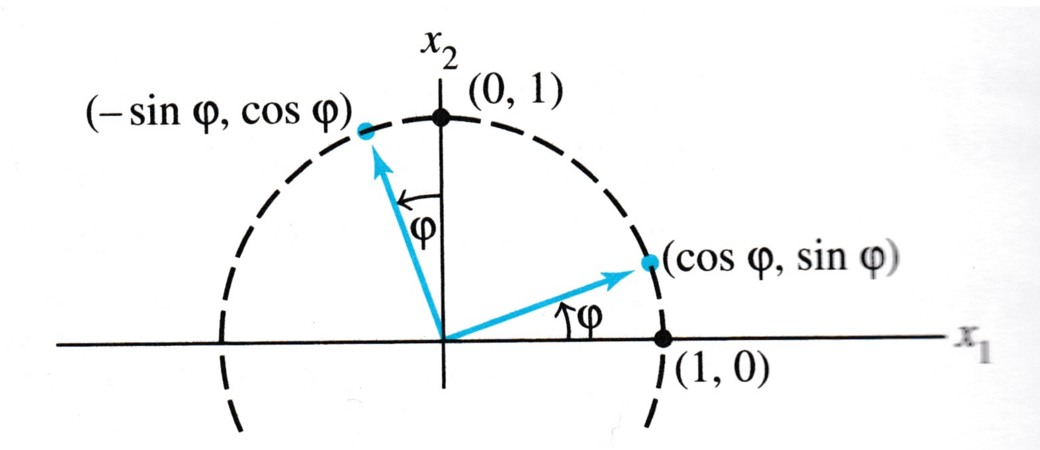
Example. Let $T : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ be the transformation that rotates each point in $\mathbb{R}^2$ about the origin through an angle φ, with counterclockwise rotation for a positive angle. Find the standard matrix A of this transformation.

Solution. The columns of I are $\mathbf{e_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\mathbf{e_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

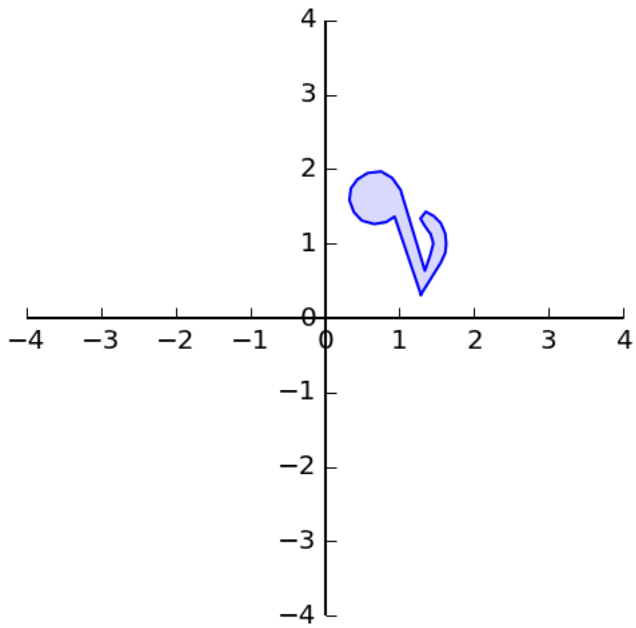
Referring to the diagram below, we can see that $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ rotates into $\begin{bmatrix} \cos\varphi \\ \sin\varphi \end{bmatrix}$, and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ rotates into $\begin{bmatrix} -\sin\varphi \\ \cos\varphi \end{bmatrix}$.

So by the Theorem above,

$$A = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}.$$

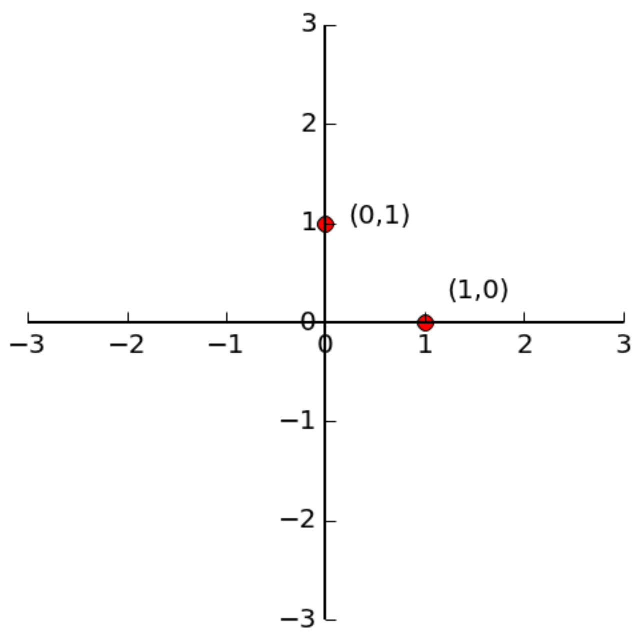



```
In [8]: ax = dm.plotSetup(-4,4,-4,4)
        note = dm.mnote()
        angle = 0.0
        phi = (angle/360.0) * 2.0 * np.pi
        rotate = np.array([[np.cos(phi), -np.sin(phi)],[np.sin(phi), np.cos(phi)]])
        rnote = rotate.dot(note)
        # reflect = np.array([[-1,0],[0,1]])
        dm.plotShape(rnote)
```



1.3 Geometric Linear Transformations of $\mathbb{R}^2$

Geometrically, we find the standard matrix of a linear tranformation of $\mathbb{R}^2$ by asking what the transfomation does to two particular points:



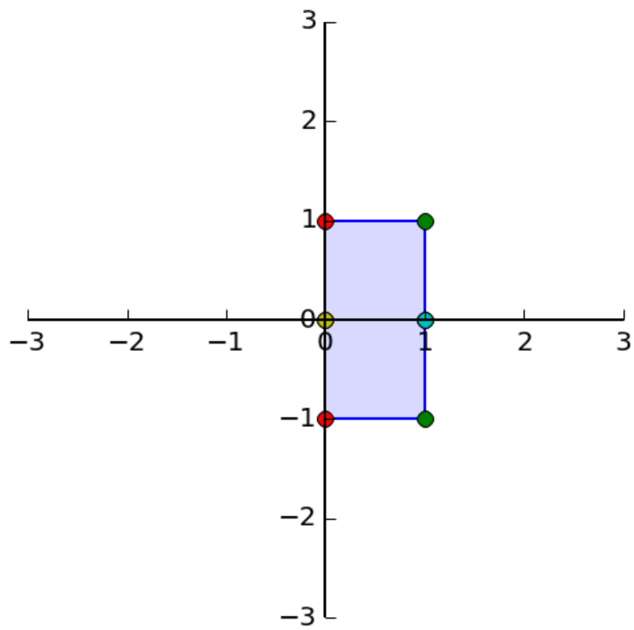
Let's look at some linear transformations of $\mathbb{R}^2$ to $\mathbb{R}^2$.

In [10]:
```
A = np.array([[1,0],[0,-1]])
print A
ax = dm.plotSetup(-3,3,-3,3)
dm.plotSquare(square)
dm.plotSquare(A.dot(square))
Latex(r'Reflection through the $x_1$ axis')
```

```
[[ 1  0]
 [ 0 -1]]
```

Out[10]:

Reflection through the x_1 axis

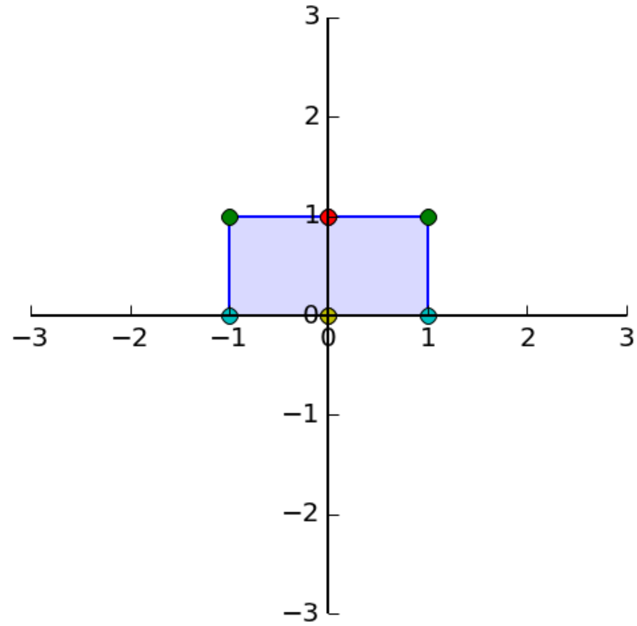


In [11]:
```
A = np.array([[-1,0],[0,1]])
print A
ax = dm.plotSetup(-3,3,-3,3)
dm.plotSquare(square)
dm.plotSquare(A.dot(square))
Latex(r'Reflection through the $x_2$ axis')
```

```
[[-1  0]
 [ 0  1]]
```

Out[11]:

Reflection through the x_2 axis

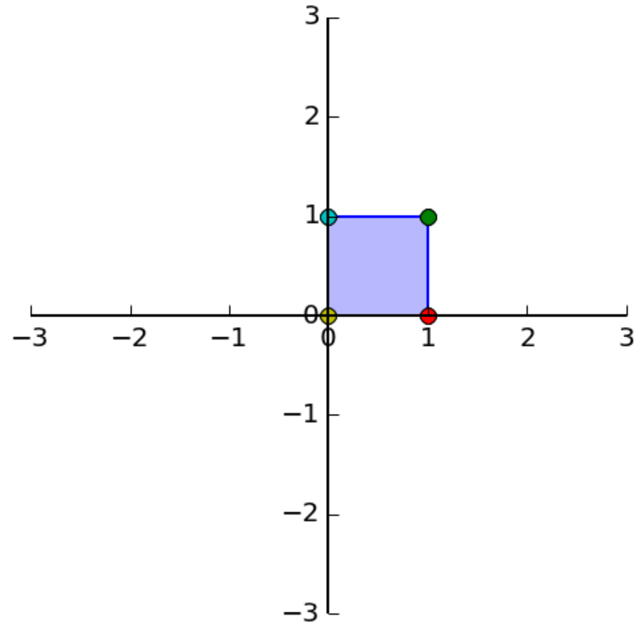



```
In [12]: A = np.array([[0,1],[1,0]])
         print A
         ax = dm.plotSetup(-3,3,-3,3)
         dm.plotSquare(square)
         dm.plotSquare(A.dot(square))
         Latex(r'Reflection through the line $x_1 = x_2$')
```

```
[[0 1]
 [1 0]]
```

Out[12]:

Reflection through the line $x_1 = x_2$

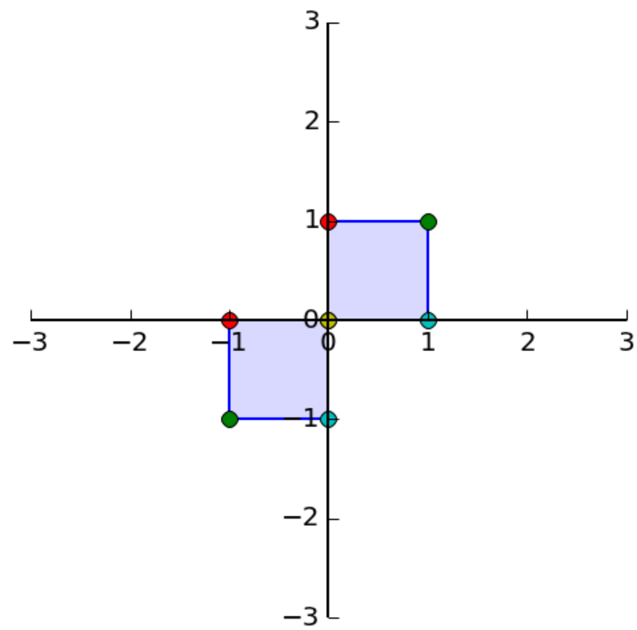


In [13]:
```
A = np.array([[0,-1],[-1,0]])
print A
ax = dm.plotSetup(-3,3,-3,3)
dm.plotSquare(square)
dm.plotSquare(A.dot(square))
Latex(r'Reflection through the line $x_1 = -x_2$')
```

```
[[ 0 -1]
 [-1  0]]
```

Out[13]:

Reflection through the line $x_1 = -x_2$

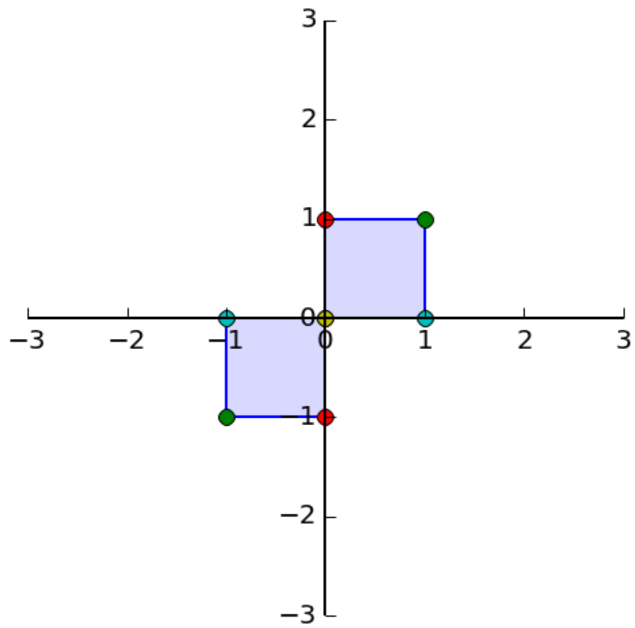


In [14]:
```
A = np.array([[-1,0],[0,-1]])
print A
ax = dm.plotSetup(-3,3,-3,3)
dm.plotSquare(square)
dm.plotSquare(A.dot(square))
Latex(r'Reflection through the origin')
```

```
[[-1  0]
 [ 0 -1]]
```

Out[14]:

Reflection through the origin

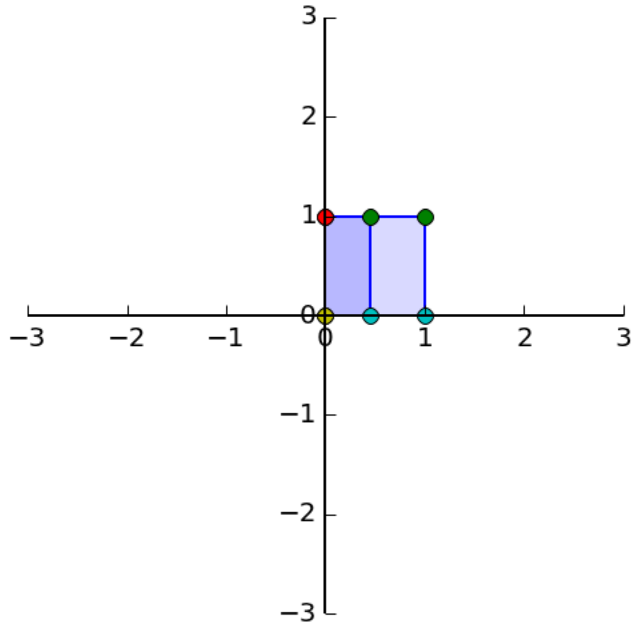



```
In [15]: A = np.array([[0.45,0],[0,1]])
         print A
         ax = dm.plotSetup(-3,3,-3,3)
         dm.plotSquare(square)
         dm.plotSquare(A.dot(square))
         Latex(r'Horizontal Contraction')
```

```
[[ 0.45  0.  ]
 [ 0.    1.  ]]
```

Out[15]:

Horizontal Contraction

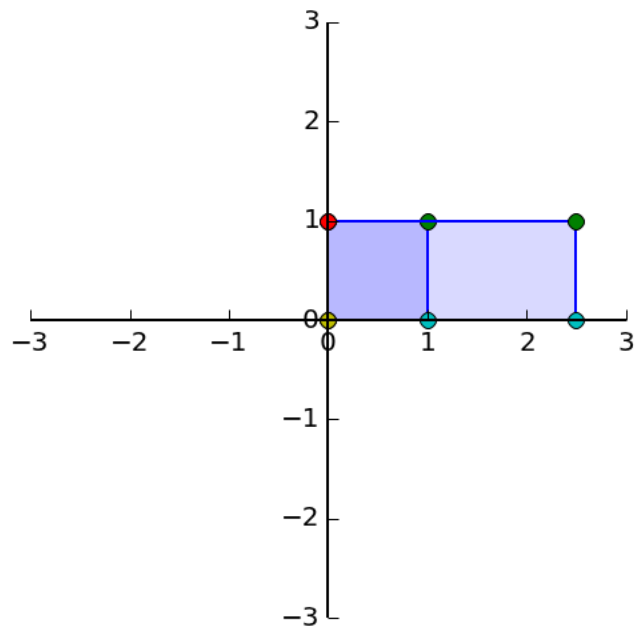



```
In [16]: A = np.array([[2.5,0],[0,1]])
         print A
         ax = dm.plotSetup(-3,3,-3,3)
         dm.plotSquare(square)
         dm.plotSquare(A.dot(square))
         Latex(r'Horizontal Expansion')
```

```
[[ 2.5  0. ]
 [ 0.   1. ]]
```

Out[16]:

Horizontal Expansion

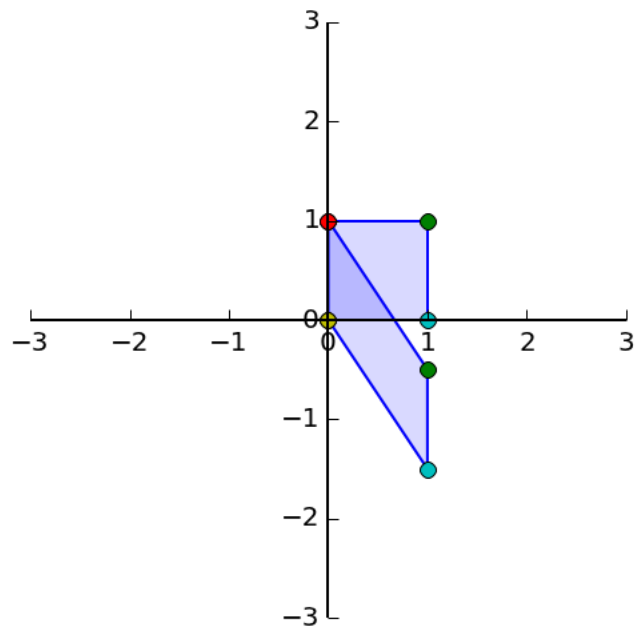



```
In [17]: A = np.array([[1,0],[-1.5,1]])
         print A
         ax = dm.plotSetup(-3,3,-3,3)
         dm.plotSquare(square)
         dm.plotSquare(A.dot(square))
         Latex(r'Vertical Shear')
```

```
[[ 1.   0. ]
 [-1.5  1. ]]
```

Out[17]:

Vertical Shear

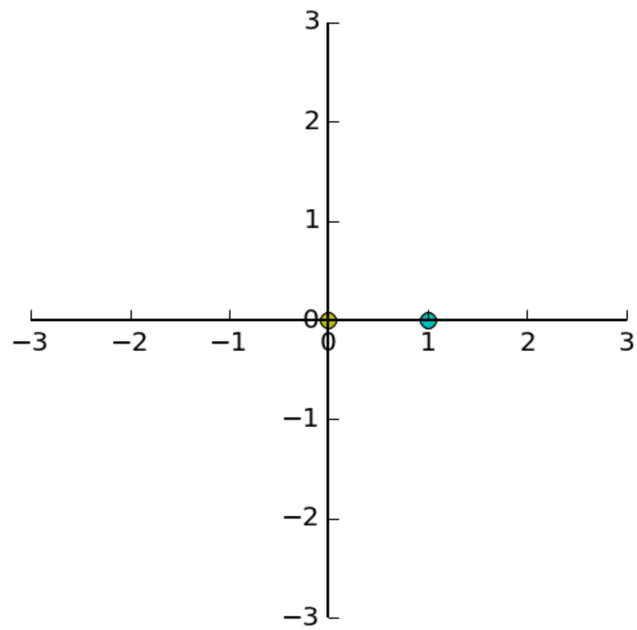



```
In [18]: A = np.array([[1,0],[0,0]])
         print A
         ax = dm.plotSetup(-3,3,-3,3)
         # dm.plotSquare(square)
         dm.plotSquare(A.dot(square))
         Latex(r'Projection onto the $x_1$ axis')
```

```
[[1 0]
 [0 0]]
```

Out[18]:

Projection onto the x_1 axis

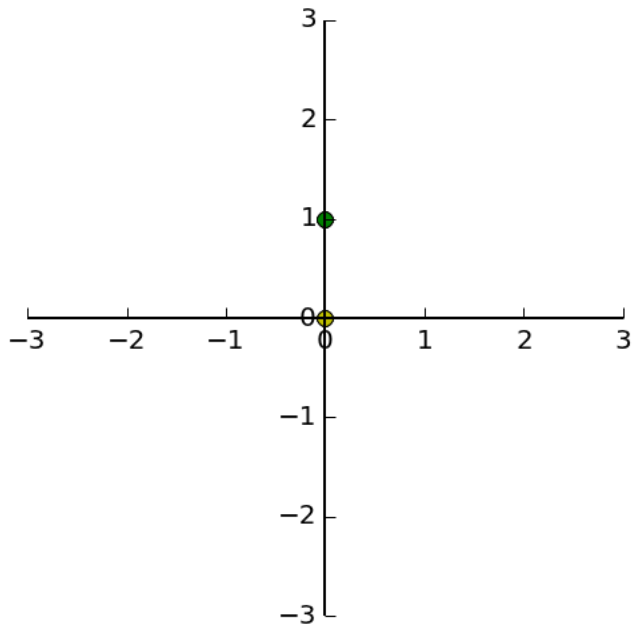



```
In [19]: A = np.array([[0,0],[0,1]])
         print A
         ax = dm.plotSetup(-3,3,-3,3)
         # dm.plotSquare(square)
         dm.plotSquare(A.dot(square))
         Latex(r'Projection onto the $x_2$ axis')
```

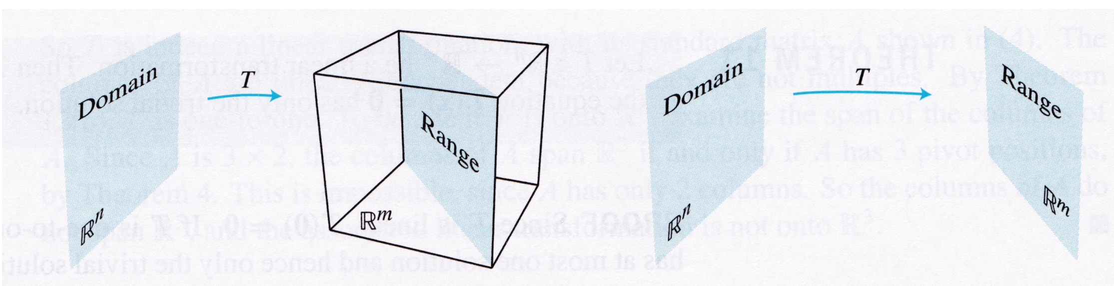
```
[[0 0]
 [0 1]]
```

Out[19]:

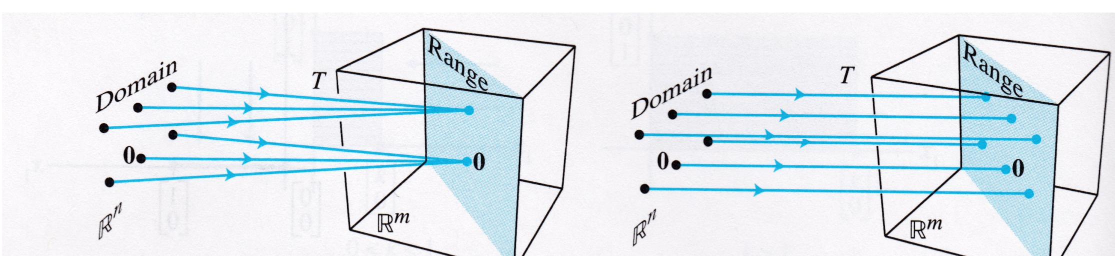
Projection onto the x_2 axis



1.4 Onto and One-to-One



T is *not* onto $\mathbb{R}^m$ *T is* onto $\mathbb{R}^m$



T is *not* one-to-one   *T is* one-to-one

```
In [21]:
```