

L6LinearTransformations

February 20, 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

$$x_1\mathbf{a}_1 + \cdots + x_n\mathbf{a}_n = \mathbf{b}.$$

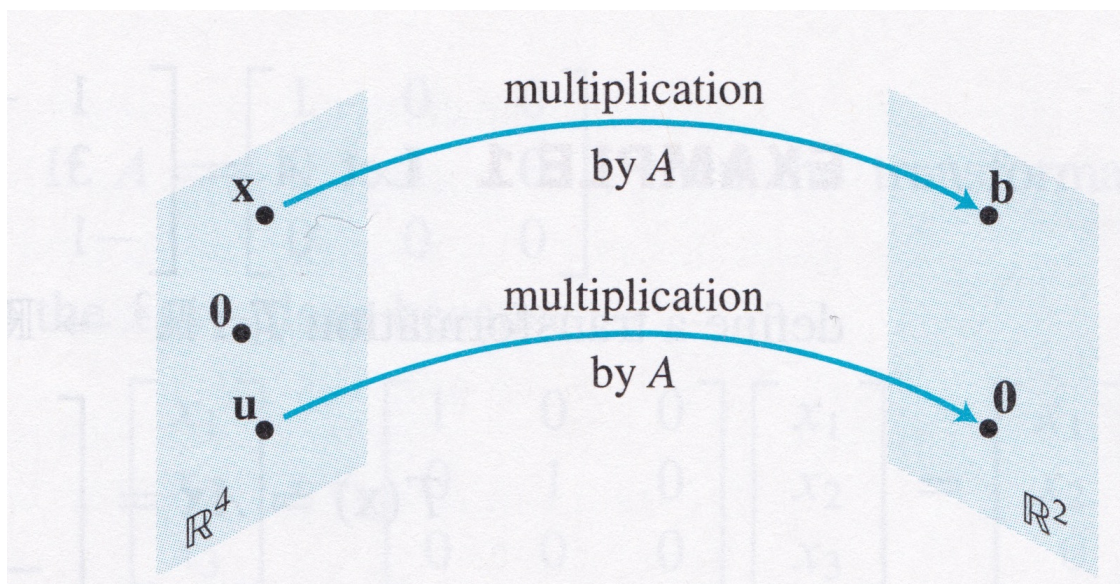
However, we'll now think of the matrix equation in a new way: we will think of A as "acting on" the vector \mathbf{x} to form a new vector \mathbf{b} .

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, functions that 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 .

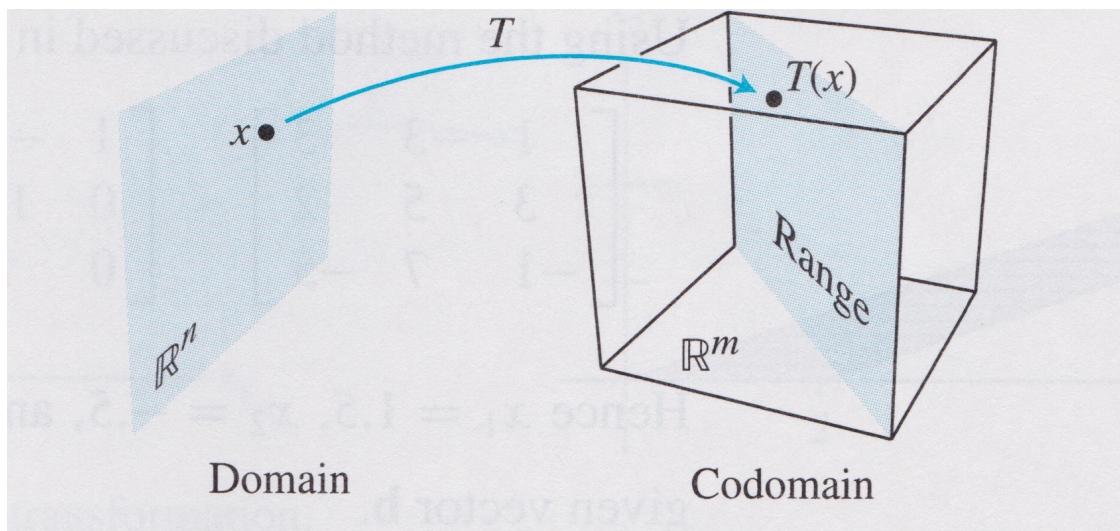
The set \mathbb{R}^n is called the **domain** of T , and \mathbb{R}^m is called the **codomain** of T .

The notation:

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

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 .



1.1 Question Time! Q6.1

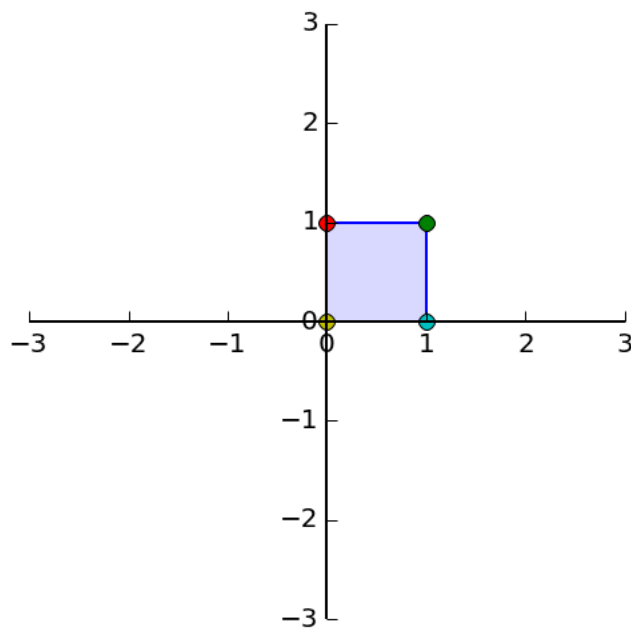
Let's do an example. Let's say I have these points in \mathbb{R}^2 :

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

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

$$A = \begin{bmatrix} 1 & 1.5 \\ 0 & 1 \end{bmatrix}.$$

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

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

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

$$A \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 1 \end{bmatrix}$$

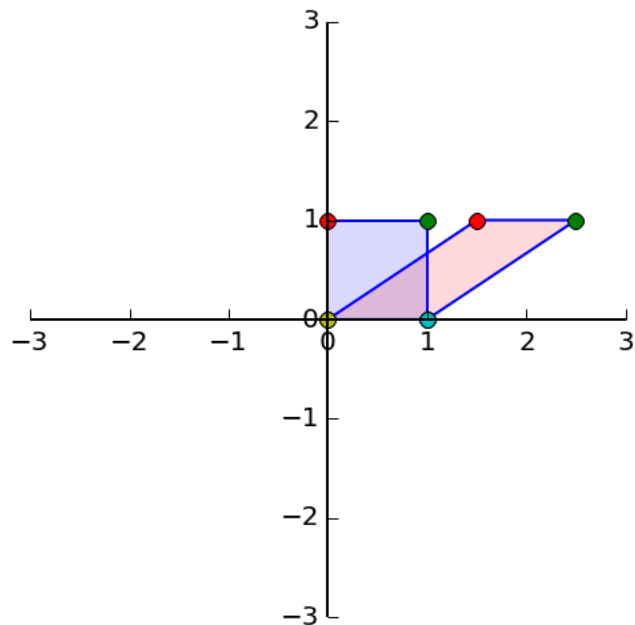
$$A \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2.5 \\ 1 \end{bmatrix}$$

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

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

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

```
square =
[[ 0.  1.  1.  0.]
 [ 1.  1.  0.  0.]]
A 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. We'll be looking at a number of examples in python.

Notice how we multiplied each column of 'square' by A .

In python/numpy, every matrix has a `.dot()` method. This method multiplies the matrix by every column of its argument, returning the results as a matrix.

That is:

`A.dot(B)`

where A and B are matrices, will multiply A by every column of B , and the resulting vectors will be formed into a matrix.

1.2 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 .

1.3 Question Time! Q6.2

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?

1.4 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 $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a linear transformation. 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 j th column is the vector $T(\mathbf{e}_j)$, where \mathbf{e}_j is the j th column of the identity matrix in \mathbb{R}^n :

$$A = [T(\mathbf{e}_1) \dots T(\mathbf{e}_n)].$$

A is called the *standard matrix* of T .

Proof. Write

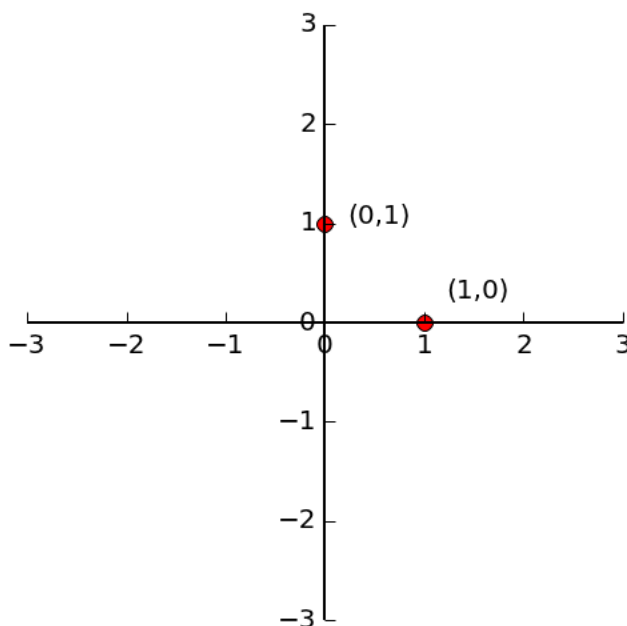
$$\mathbf{x} = I\mathbf{x} = [\mathbf{e}_1 \dots \mathbf{e}_n] \mathbf{x}$$

$$= x_1 \mathbf{e}_1 + \cdots + x_n \mathbf{e}_n.$$

Because T is linear, we have:

$$\begin{aligned} T(\mathbf{x}) &= T(x_1 \mathbf{e}_1 + \cdots + x_n \mathbf{e}_n) \\ &= x_1 T(\mathbf{e}_1) + \cdots + x_n T(\mathbf{e}_n) \\ &= [T(\mathbf{e}_1) \cdots T(\mathbf{e}_n)] \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = A\mathbf{x}. \end{aligned}$$

For example, we find the standard matrix of a linear transformation of \mathbb{R}^2 by asking what the transformation does to two particular points:



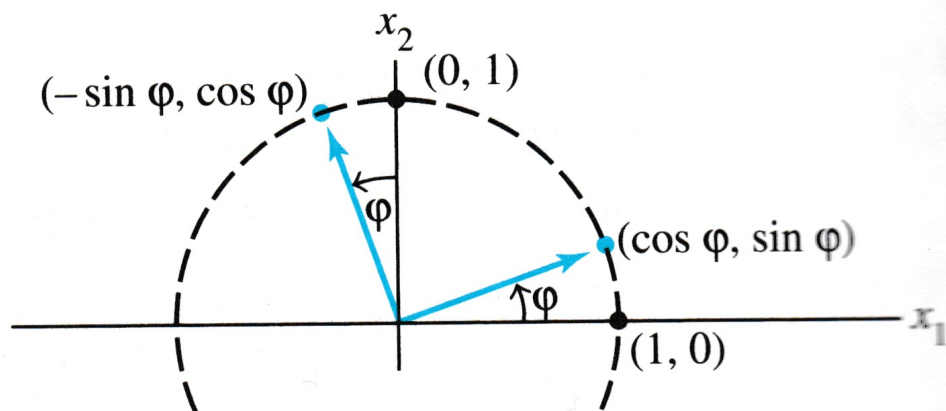
This is a hugely powerful tool. Let's say we start from a linear transformation; we can use this to find the matrix that implements that linear transformation.

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 $\mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

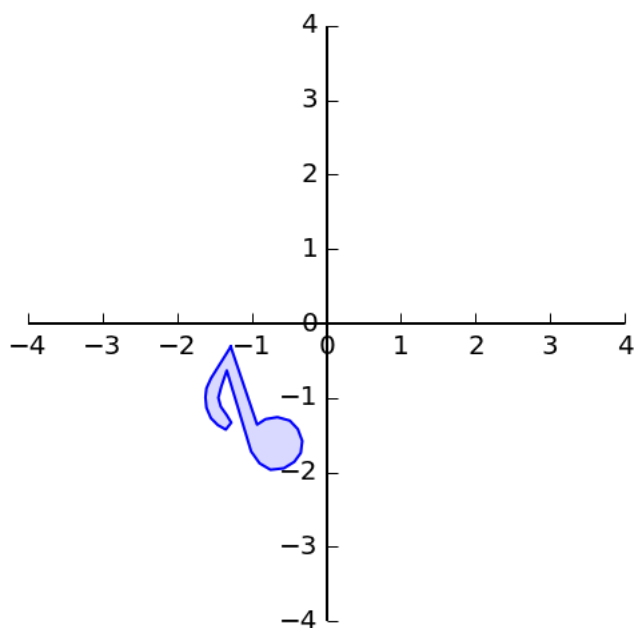
Referring to the diagram below, we can see that $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ rotates into $\begin{bmatrix} \cos \varphi \\ \sin \varphi \end{bmatrix}$, and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ rotates into $\begin{bmatrix} -\sin \varphi \\ \cos \varphi \end{bmatrix}$.



So by the Theorem above,

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

```
In [26]: ax = dm.plotSetup(-4,4,-4,4)
note = dm.mnote()
angle = 180.0; 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)
dm.plotShape(rnote)
```



1.5 Question Time! Q6.3

1.6 Geometric Linear Transformations of \mathbb{R}^2

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

```
In [10]: A = np.array([[2.5,0],[0,2.5]])
print A
# ax = dm.plotSetup(-3,3,-3,3)
# dm.plotSquare(square)
# dm.plotSquare(A.dot(square),'r')
```

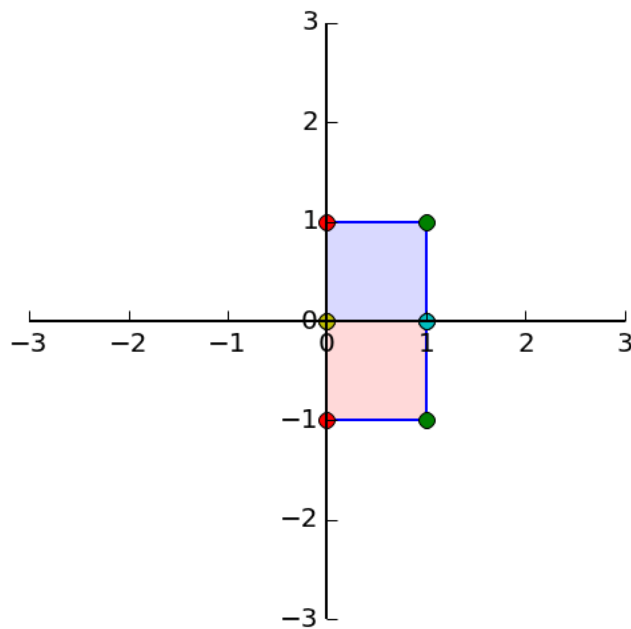
```
[[ 2.5  0. ]
 [ 0.  2.5]]
```

```
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),'r')
Latex(r'Reflection through the  $x_1$  axis')
```

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

Out[11]:

Reflection through the x_1 axis



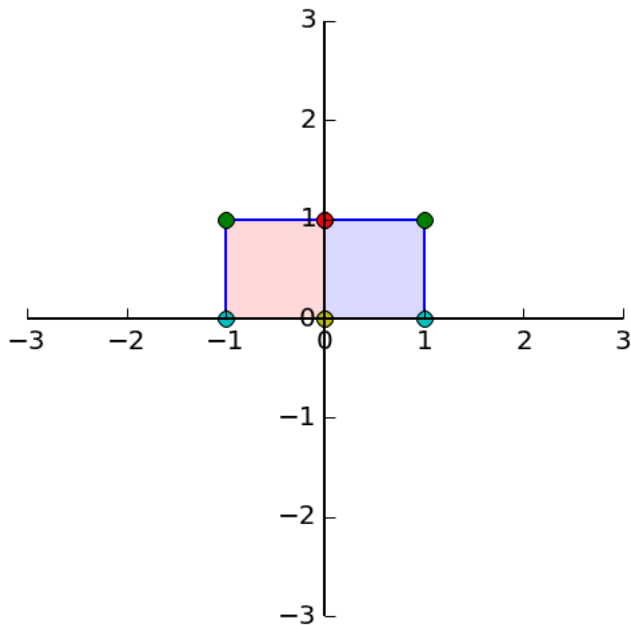
```
In [12]: A = np.array([[ -1,0],[0,1]])
print A
ax = dm.plotSetup(-3,3,-3,3)
dm.plotSquare(square)
dm.plotSquare(A.dot(square),'r')
Latex(r'Reflection through the  $x_2$  axis')
```



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

Out[12]:

Reflection through the x_2 axis



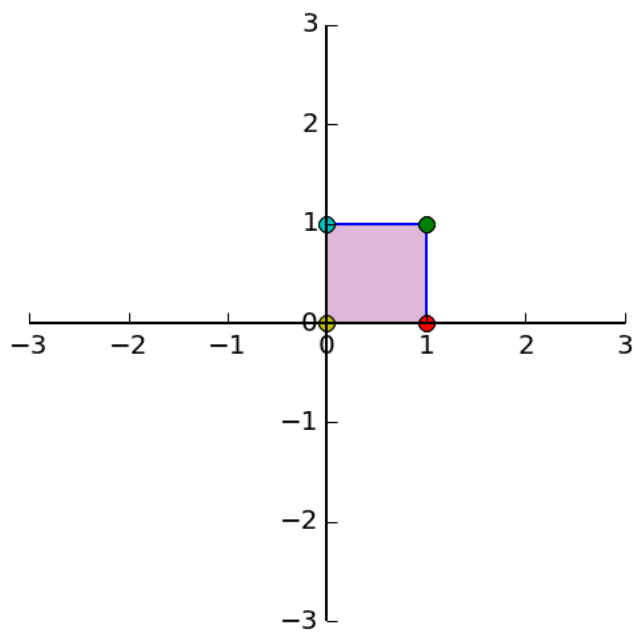
```
In [13]: A = np.array([[0,1],[1,0]])
print A
ax = dm.plotSetup(-3,3,-3,3)
dm.plotSquare(square)
dm.plotSquare(A.dot(square),'r')
Latex(r'Reflection through the line  $x_1 = x_2$ ')

```

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

Out[13]:

Reflection through the line $x_1 = x_2$



```
In [14]: A = np.array([[0,-1],[-1,0]])
print A
ax = dm.plotSetup(-3,3,-3,3)
dm.plotSquare(square)
dm.plotSquare(A.dot(square),'r')
Latex(r'Reflection through the line  $x_1 = -x_2$ ')

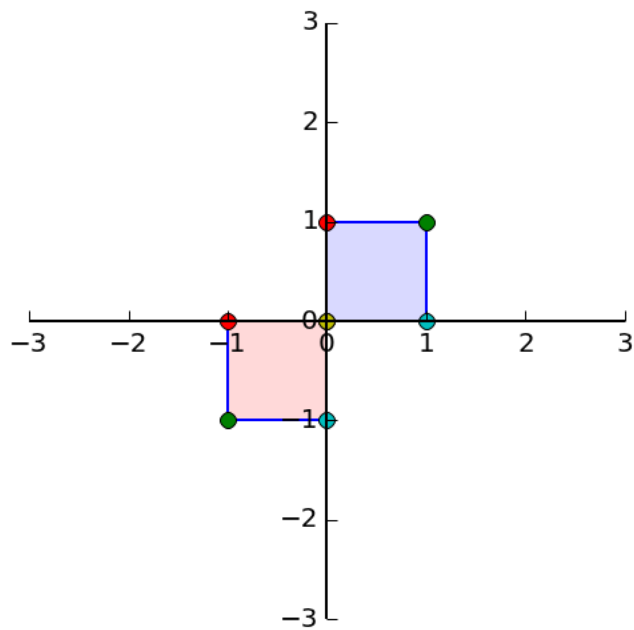
```

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

```

```
Out[14]:
Reflection through the line  $x_1 = -x_2$ 

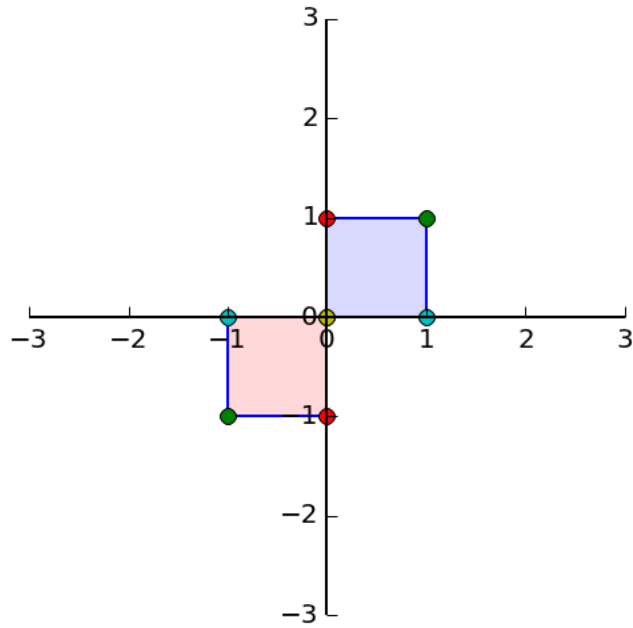
```



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

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

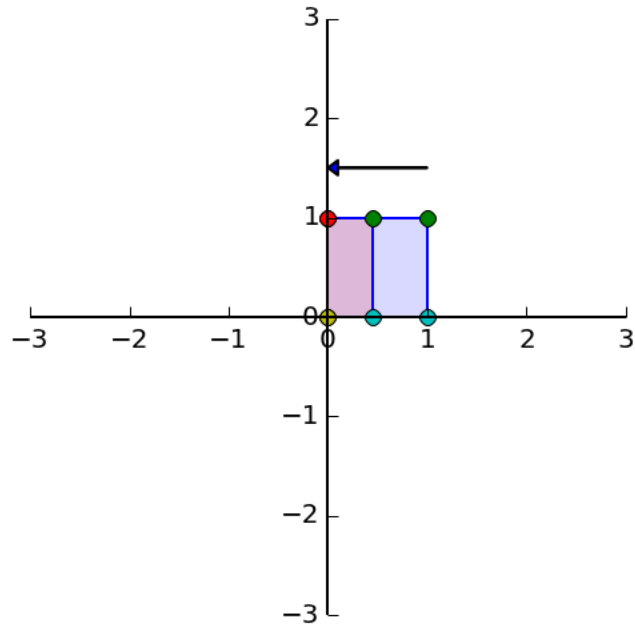
```
Out[15]:
Reflection through the origin
```



```
In [16]: 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),'r')
ax.arrow(1.0,1.5,-1.0,0,head_width=0.15, head_length=0.1, length_includes_head=True)
Latex(r'Horizontal Contraction')
```

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

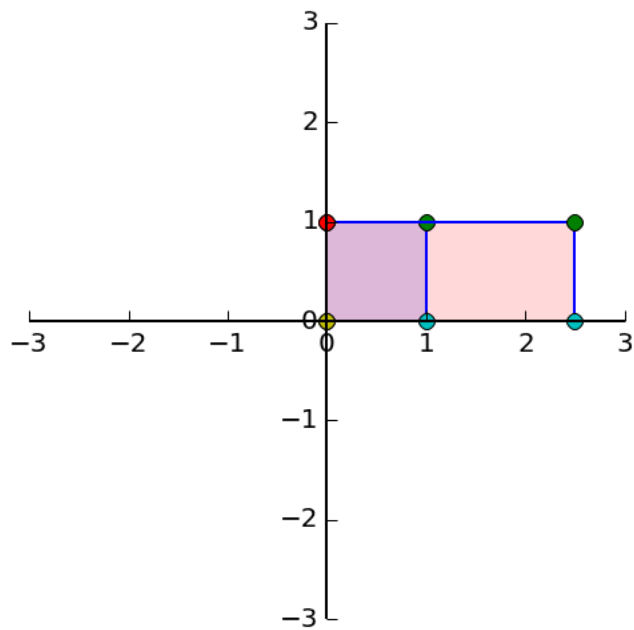
```
Out[16]:
Horizontal Contraction
```



```
In [17]: 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),'r')
Latex(r'Horizontal Expansion')
```

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

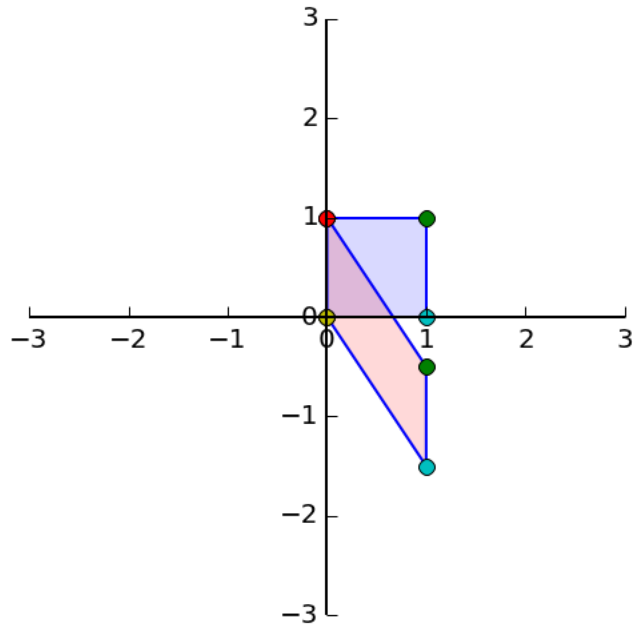
```
Out[17]:
Horizontal Expansion
```



```
In [18]: 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),'r')
          Latex(r'Vertical Shear')
```

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

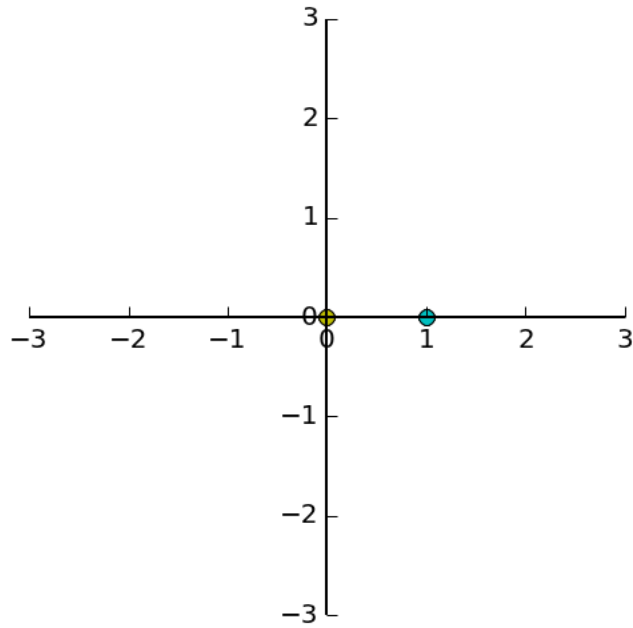
```
Out[18]:
Vertical Shear
```



```
In [19]: A = np.array([[1,0],[0,0]])
print A
ax = dm.plotSetup(-3,3,-3,3)
# dm.plotSquare(square)
dm.plotSquare(A.dot(square),'r')
# ax.arrow(1.0,1.0,0,-0.9,head_width=0.15, head_length=0.1, length_includes_head=True)
# ax.arrow(0.0,1.0,0,-0.9,head_width=0.15, head_length=0.1, length_includes_head=True)
Latex(r'Projection onto the  $x_1$  axis')
```

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

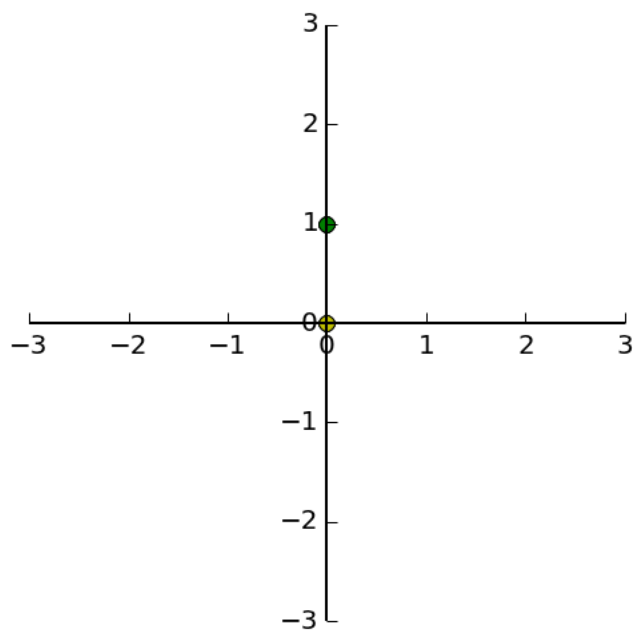
```
Out[19]:
Projection onto the  $x_1$  axis
```



```
In [20]: A = np.array([[0,0],[0,1]])
          print A
          ax = dm.plotSetup(-3,3,-3,3)
          # dm.plotSquare(square)
          dm.plotSquare(A.dot(square))
          #ax.arrow(1.0,1.0,-0.9,0,head_width=0.15, head_length=0.1, length_includes_head=True)
          #ax.arrow(1.0,0.0,-0.9,0,head_width=0.15, head_length=0.1, length_includes_head=True)
          Latex(r'Projection onto the  $x_2$  axis')
```

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

```
Out[20]:
Projection onto the  $x_2$  axis
```

1.7 Onto and One-to-One

