Matrix Manipulation Worksheet

Start by loading the linalg (linear algebra) package to run matrix formula (don't worry about the warning message.)

```
> restart;
  with(linalg):
Warning, the protected names norm and trace have been redefined and
unprotected
```

Create our matrix with which we will manipulate our image. In this example, let's double the height of the image and flip it across the y-axis. For this, (1,0) will move to (-1,0) and (0,1) will move to (0,2). Therefore, the matrix is as below.

```
> A:=matrix([[-1,0],[0,2]]);
```

$$A := \begin{bmatrix} -1 & 0 \\ 0 & 2 \end{bmatrix}$$

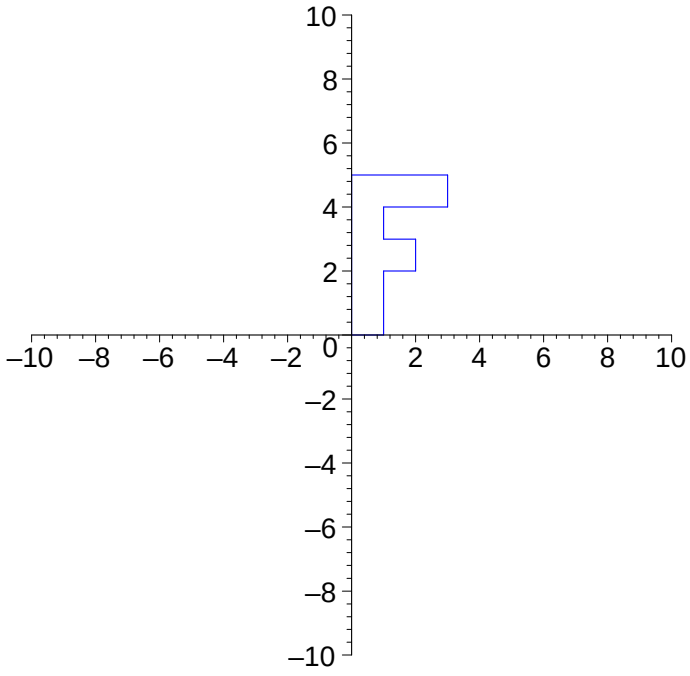
Let's set up a list of lists, which can be plotted in Maple. Remember, this is an object of the type listlist, or a list of smaller lists (in this case, we have lists with two elements, the coordinates).

```
> f1 := [[0,0], [0,5], [3,5], [3,4], [1,4], [1,3],
        [2,3], [2,2], [1,2], [1,0], [0,0]];
```

$$f1 := [[0, 0], [0, 5], [3, 5], [3, 4], [1, 4], [1, 3], [2, 3], [2, 2], [1, 2], [1, 0], [0, 0]]$$

Here's how you plot a list of lists. The style=LINE draws lines between subsequent points. We duplicated the first point at the end so that the last line is drawn.

```
> plot(f1, style=LINE, color=blue, thickness=2, view=[-10..10,
  -10..10], scaling=constrained);
```



Now the tricky part. In order to manipulate each point, we need to conver the listlist to an

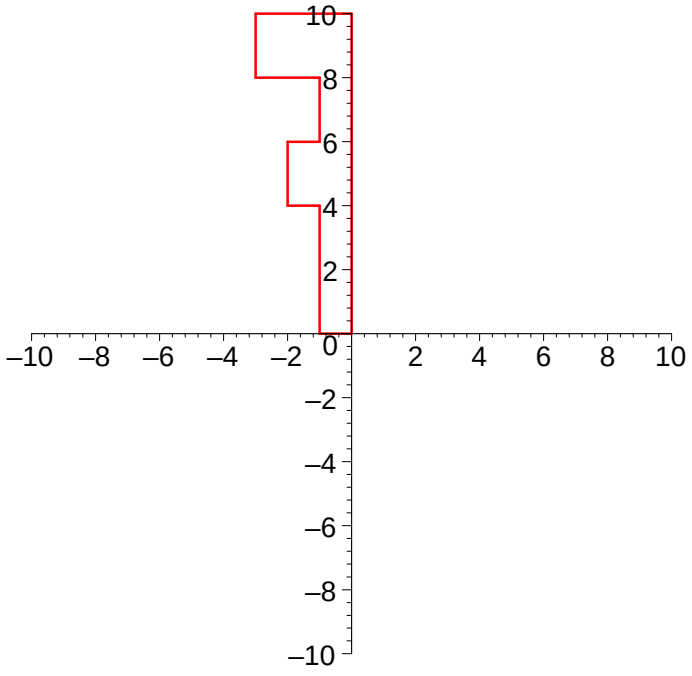
array (basically a large matrix). After we evalulate the matrix multiplication (evalm(...)) then we convert the result back into a listlist. This is the way I discovered; I hope for the sake of Maple usability that there's an easier way. For now, just use this.

```
> f2:=convert(evalm(convert(f1, array) &* A), listlist);
```

$$f2 := [[0, 0], [0, 10], [-3, 10], [-3, 8], [-1, 8], [-1, 6], [-2, 6], [-2, 4], [-1, 4], [-1, 0], [0, 0]]$$

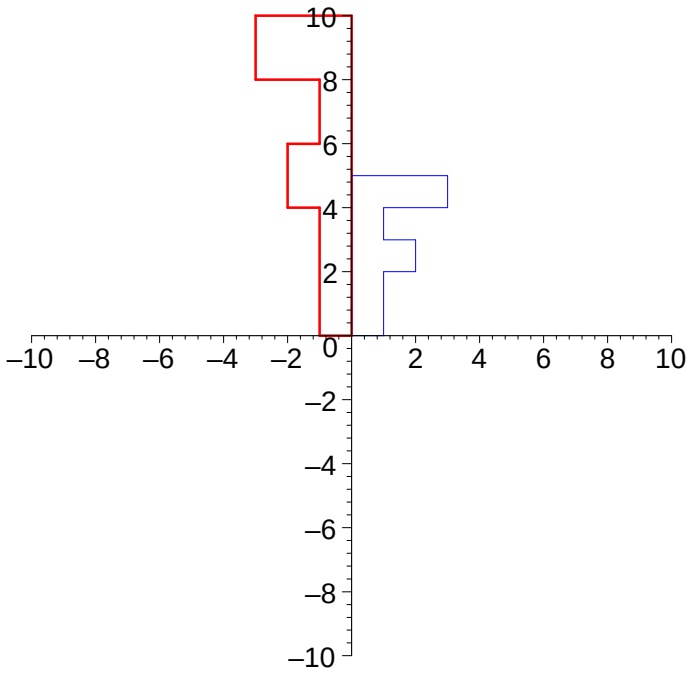
...and plotting the resulting set of points gives is what we desired in the first place--the image stretched vertically by a factor of two and flipped across the y-axis.

```
> plot(f2, style=LINE, color=red, thickness=3, view=[-10..10,
  -10..10], scaling=constrained);
```



If we so desired, we could plot both at the same time! We simply set it up as so:

```
> plot({f1, f2}, color=[blue, red], thickness=[1,3],
  view=[-10..10, -10..10], scaling=constrained);
```

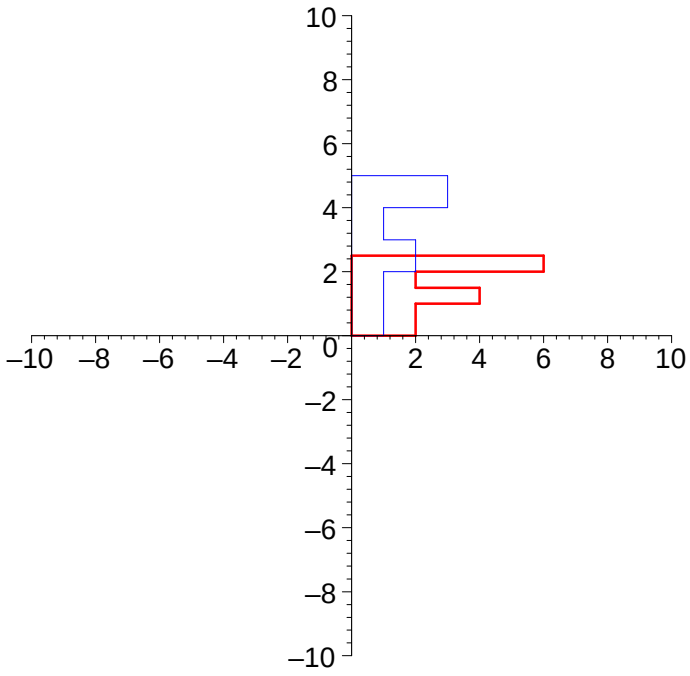


Other examples:

Stretch matrix by two in horizontal direction and shrink by two in vertical direction

```
> A3 := matrix([[2,0],[0,1/2]]);
  f3:=convert(evalm(convert(f1, array) &* A3), listlist):
  plot({f1,f3}, style=LINE, color=[blue,red], thickness=[1,3],
  view=[-10..10, -10..10], scaling=constrained);
```

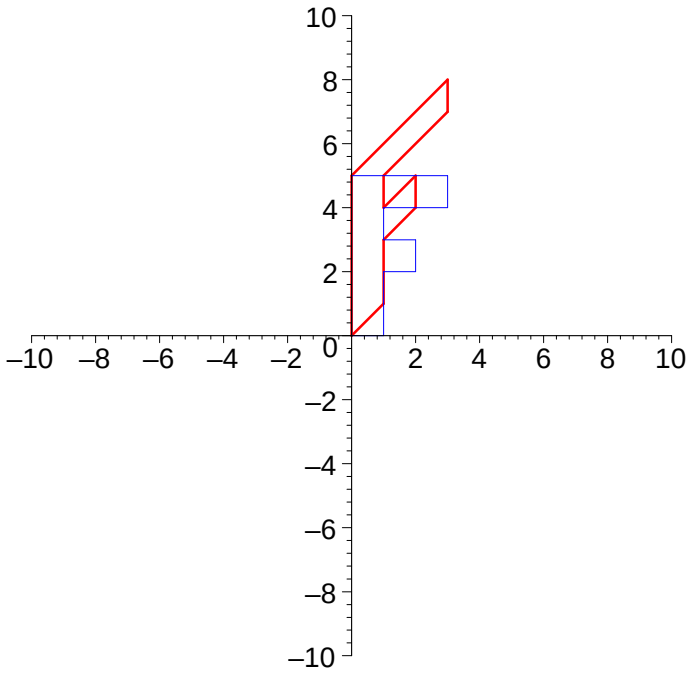
$$A3 := \begin{bmatrix} 2 & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$



Skew image - send (1,0) to (1,1) and (0,1) to (0,1)

```
> A4 := matrix([[1,1],[0,1]]);
  f4 := convert(evalm(convert(f1, array) &* A4), listlist):
  plot({f1,f4}, style=LINE, color=[blue,red], thickness=[1,3],
  view=[-10..10, -10..10], scaling=constrained);
```

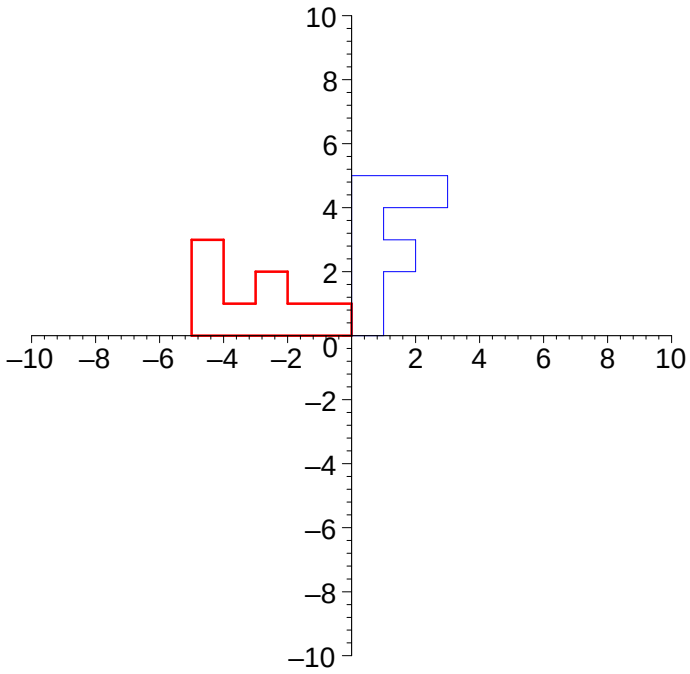
$$A4 := \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$



Rotate Image counter-clockwise 90 degrees - send (1,0) to (0,1) and (0,1) to (-1,0)

```
> A5 := matrix([[0,1], [-1,0]]);
  f5 := convert(evalm(convert(f1, array) &* A5), listlist):
  plot({f1,f5}, style=LINE, color=[blue,red], thickness=[1,3],
  view=[-10..10, -10..10], scaling=constrained);
```

$$A5 := \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$



Brian Abram
University of Pennsylvania