Background

These exercises are based upon the "Reality Check" of our textbook, which includes the control points for several vector fonts. Each letter is described by an $N \times 8$ matrix, where each row of the matrix gives the coordinates of the endpoints and control points of one curve. These matrices are available on the course website as `t.m` and `f.m`. You may find it helpful to use `beziercoeff.m` (also available on the course website) to compute coefficients of any one curve.

You may find the following MATLAB commands helpful for these exercises:

- `clf;` `% Clears the plotting window`
- `hold on;` `% Allows subsequent plots to be superimposed`
- `axis equal;` `% Make scaling for x and y axis the same`
- `set(gca, 'xtick', []);` `% Removes tick marks from the x-axis`
- `set(gca, 'ytick', []);` `%  "  "  "  "  "  y-axis`

Exercise 1. Design and implement a MATLAB function, `bezierplot(M)`, which will plot the Bézier curves described by the matrix `M`.
The MATLAB commands

```
load f.m
load t.m
```

will create matrices named `f` and `t`, which you can use to illustrate your code.

Solution.

Exercise 2. This exercise asks you to generalize the `bezierplot` function. You might remember the concept of a "rotation matrix" from linear algebra. In two dimensions, this matrix is:

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

A rotation matrix has the effect of rotating a vector counterclockwise by the angle θ. In other words, the point (x, y) can be transformed to a new point (x', y') by a simple matrix multiplication:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

By applying the rotation matrix to every endpoint and control point, we can easily display a composite Bézier curve at any desired angle. For an example, see Figure 1.

Design and implement a MATLAB function, `rotbezierplot(M, theta)`, which displays the composite Bézier curve described by `M`, rotated about the origin by `theta` degrees.

Be sure that your code accounts for the conversion from degrees to radians, since the trigonometric functions expect the argument in radians. Since everybody knows (right?) 2π radians is 360 degrees, the conversion is easily accomplished.

You should be able to reproduce Figure 1, but you may generate any output which convinces me that your code can produce rotated, composite curves.

Solution.

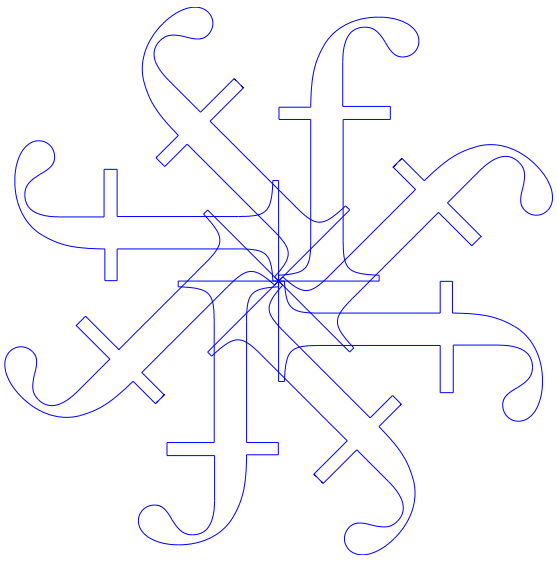

Figure 1: Eight composite Bézier curves, rotated about the origin. Each letter consists of a 21-piece curve. This figure was obtained by invoking `rotbezierplot` eight times, with various angles.