Exercise 2

Unknown Author

February 25, 2014

1 Exercise 2: Finding Hermite Curve Parameters

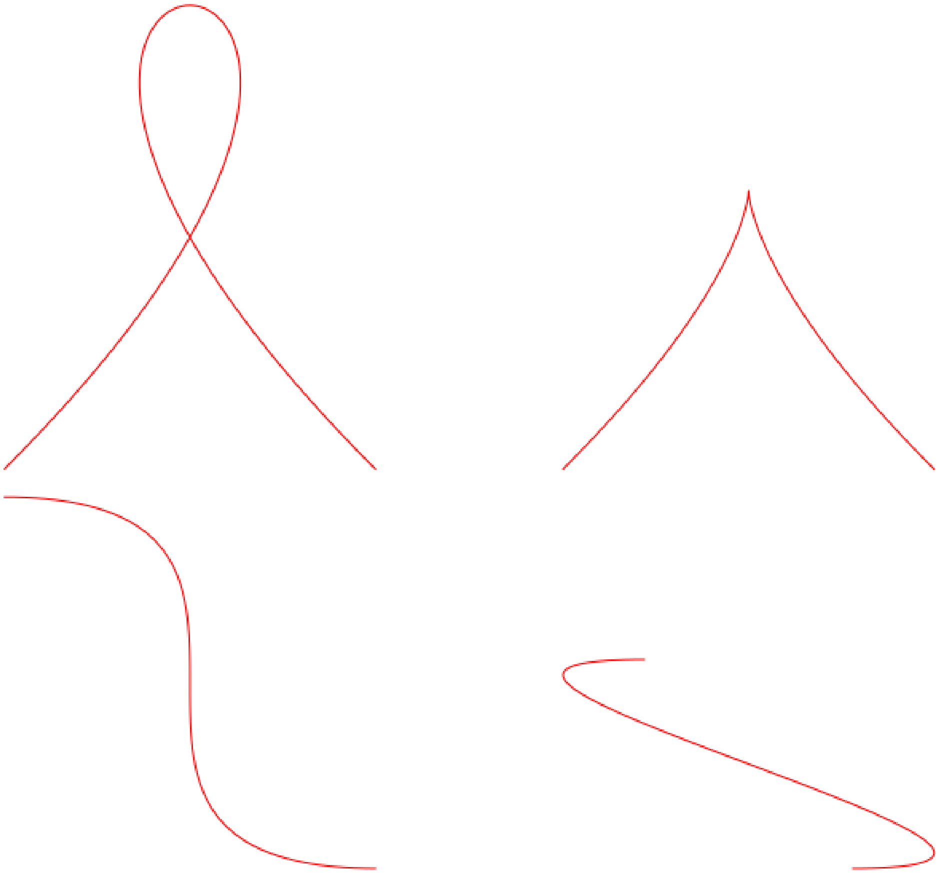
Find the values of the points P 0 and P 1 as the values of the derivatives at these endpoints to draw the four curves below:

In [1]:

```
from IPython.display import Image

Image(filename='img_ex_2.png')
```

Out [1]:



In [46]:

```
%matplotlib inline
import numpy
import numpy as np
import pylab
import matplotlib.pyplot as plt

class HermiteCurve:

    def __init__(self):
        self.p0 = np.array([0, 0])
        self.p1 = np.array([0, 0])
        self.p0_tangent = np.array([0, 0])
        self.p1_tangent = np.array([0, 0])
        self.plt = plt

        self.calculcate_coefficients()
        self.setup_plot()

    def setup_plot(self):
        self.plt.margins(0.1)
        self.plt.grid()

    def calculcate_coefficients(self):
        self.a = 2 * self.p0 - 2 * self.p1 + self.p0_tangent + self.p1_tangent
        self.b = -3 * self.p0 + 3 * self.p1 - 2 * self.p0_tangent - self.p1_tangent
        self.c = self.p0_tangent
        self.d = self.p0

    def calculate_coefficient_matrix(self):
        coefficients = numpy.matrix(
            (
                ( 2, -2,  1,  1),
                (-3,  3, -2, -1),
                ( 0,  0,  1,  0),
                ( 1,  0,  0,  0)
            )
        )

        points = numpy.matrix(
            (
                (self.p0),
                (self.p1),
                (self.p0_tangent),
                (self.p1_tangent)
            )
        )

        return np.dot(coefficients, points)

    def calculate_curve_point(self, time):
        return self.a * np.power(time, 3) + self.b * np.power(time, 2) + self.c * time

    def plot(self):
        # Plot points
        x = [self.p0[0], self.p1[0]]
        y = [self.p0[1], self.p1[1]]
        # Plot arrow
        plt.arrow(
            self.p0[0],
            self.p0[1],
            self.p0_tangent[0] - self.p0[0],
            self.p0_tangent[1] - self.p0[1]
        )
        plt.arrow(
            self.p1[0],
            self.p1[1],
            self.p1_tangent[0] - self.p1[0],
            self.p1_tangent[1] - self.p1[1]
        )
        plt.plot(x, y, 'ro')

        # Prepare data
        self.calculcate_coefficients()
        coeff_matrix = self.calculate_coefficient_matrix()

        # Gather data
        plot_data = []
        for t in numpy.linspace(0, 1, 100):
            plot_data.append(
                numpy.dot(
                    numpy.matrix(
                        (
                            (numpy.power(t, 3)),
                            (numpy.power(t, 2)),
                            (t),
                            (1)
                        )
                    ),
                    coeff_matrix
                )
            )

        x = []
        y = []
        for data in plot_data:
            x.append(data.A[0][0])
            y.append(data.A[0][1])

        # Plot
        plt.plot(x, y)
```

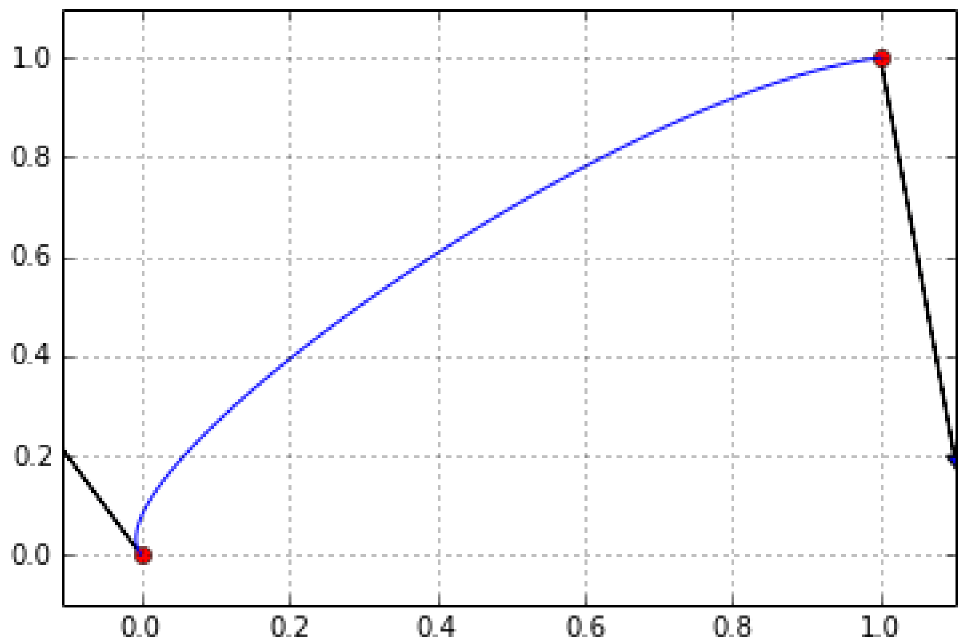
In [9]:

```
# Just some test..

c = HermiteCurve()
# P0
c.p0 = np.array([0, 0])
c.p0_tangent = np.array([-0.25, 0.5])

# P1
c.p1 = np.array([1, 1])
c.p1_tangent = np.array([1.1, 0.2])

c.plot()
```

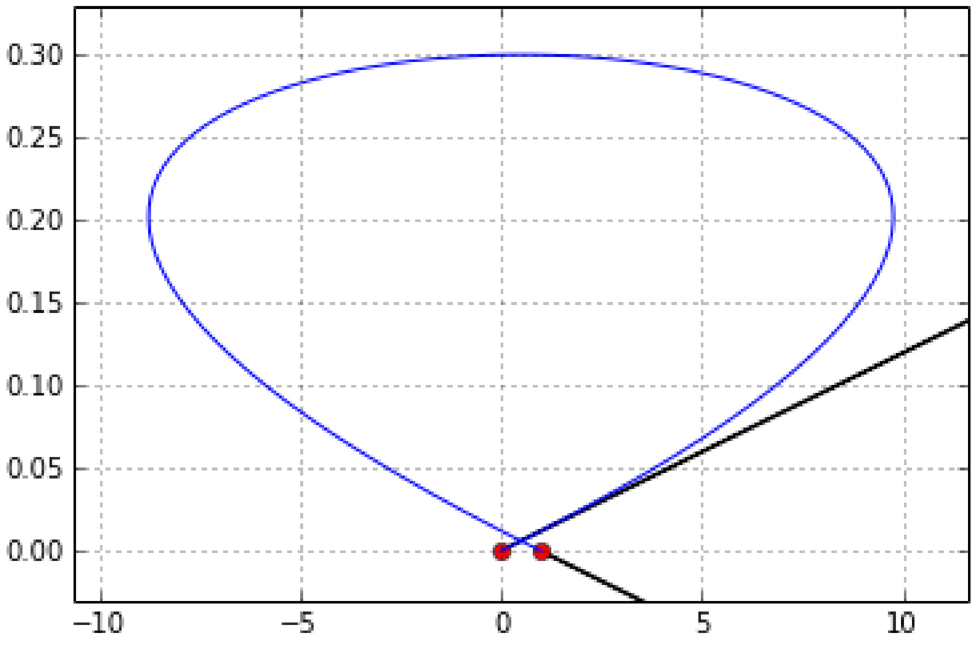


In [10]:

```
# Here we go.. 2.1
c = HermiteCurve()
# P0
c.p0 = np.array([0, 0])
c.p0_tangent = np.array([100.3, 1.2])

# P1
c.p1 = np.array([1, 0])
c.p1_tangent = np.array([100.3, -1.2])

c.plot()
```

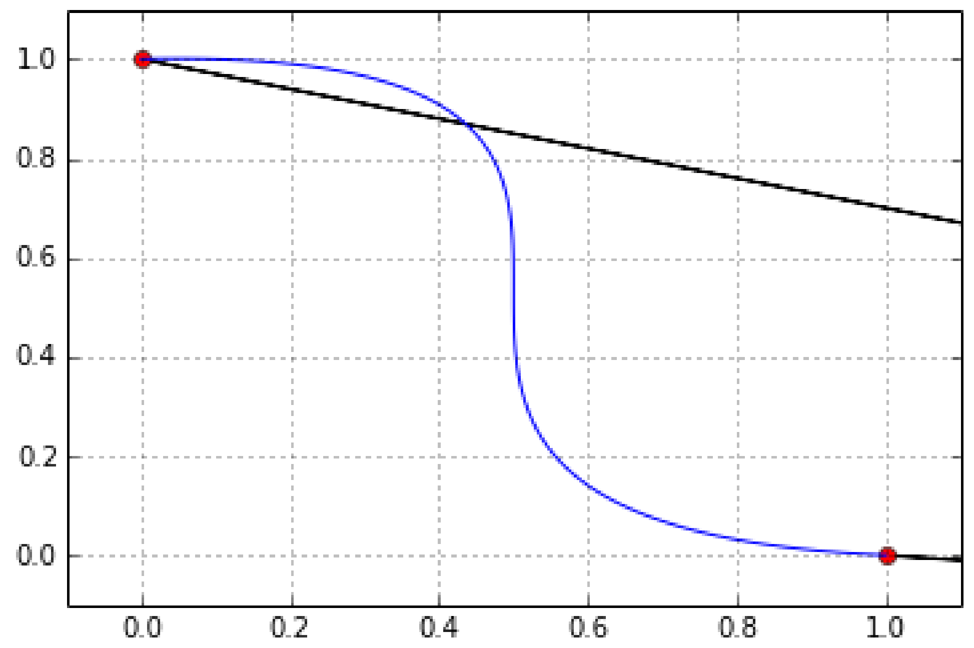


In [11]:

```
# Here we go.. 2.3
c = HermiteCurve()
# P0
c.p0 = np.array([0, 1])
c.p0_tangent = np.array([3, 0.1])

# P1
c.p1 = np.array([1, 0])
c.p1_tangent = np.array([3, -0.2])

c.plot()
```

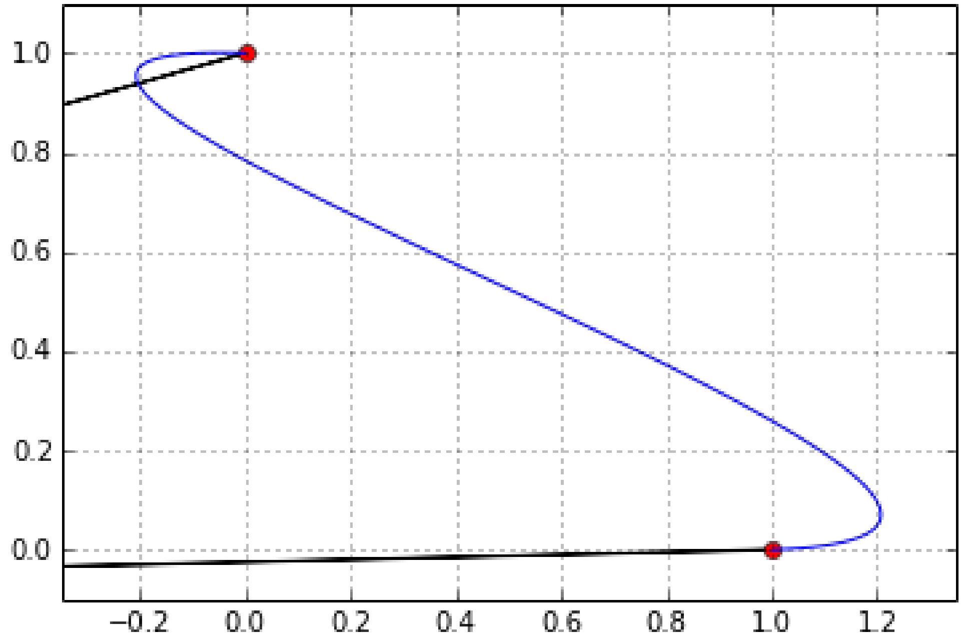


In [34]:

```
# Here we go.. 2.4
c = HermiteCurve()
# P0
c.p0 = np.array([0, 1])
c.p0_tangent = np.array([-3, 0.1])

# P1
c.p1 = np.array([1, 0])
c.p1_tangent = np.array([-3, -0.1])

c.plot()
```

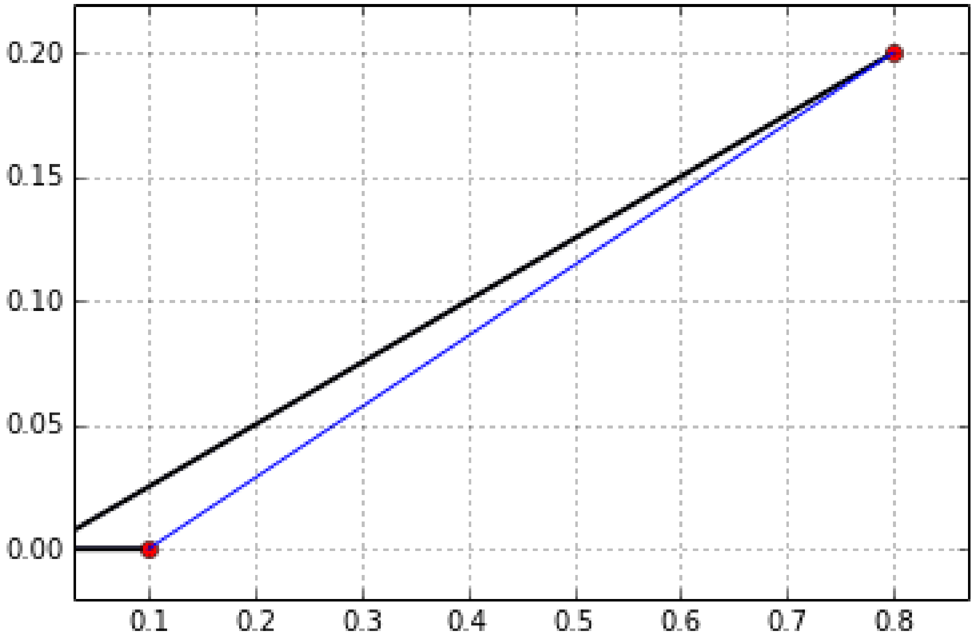


In [38]:

```
# Here we go.. 4
c = HermiteCurve()
# P0
c.p0 = np.array([0.8, 0.2])
c.p0_tangent = np.array([0,0])

# P1
c.p1 = np.array([0.1, 0])
c.p1_tangent = np.array([0,0])

c.plot()
```



In [51]:

```
# Here we go.. 5
c = HermiteCurve()
# P0
c.p0 = np.array([0, 0])
c.p0_tangent = np.array([1,0])

# P1
c.p1 = np.array([1, 1])
c.p1_tangent = np.array([0,1])

c.plot()

c2 = HermiteCurve()
# P1
c2.p0 = np.array([1, 1])
c2.p0_tangent = np.array([0,1])

# P2
c2.p1 = np.array([0, 2])
c2.p1_tangent = np.array([-1,0])

c2.plot()

c3 = HermiteCurve()
# P2
c3.p0 = np.array([0, 2])
c3.p0_tangent = np.array([-1,0])

# P3
c3.p1 = np.array([-1, 1])
c3.p1_tangent = np.array([0,-1])

c3.plot()

for u in numpy.arange(0, 1, 0.2):
    print "Curve 0 at {0}: {1}".format(u, c.calculate_curve_point(u))
    print "Curve 1 at {0}: {1}".format(u, c2.calculate_curve_point(u))
    print "Curve 2 at {0}: {1}".format(u, c3.calculate_curve_point(u))
    print ""
```

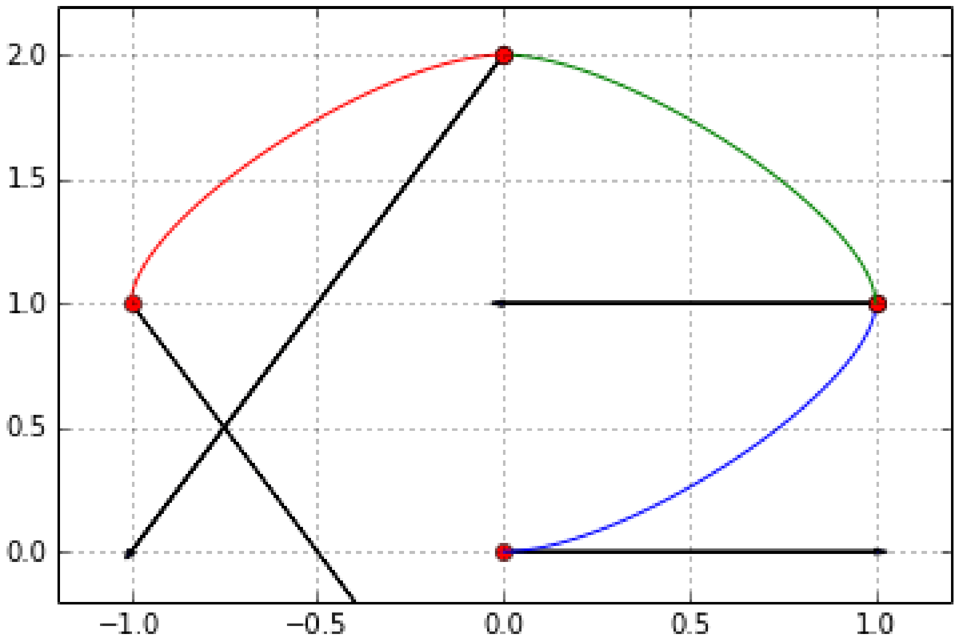
```
Curve 0 at 0.0: [ 0.  0.]
Curve 1 at 0.0: [ 1.  1.]
Curve 2 at 0.0: [ 0.  2.]

Curve 0 at 0.2: [ 0.232  0.072]
Curve 1 at 0.2: [ 0.928  1.232]
Curve 2 at 0.2: [-0.232  1.928]

Curve 0 at 0.4: [ 0.496  0.256]
Curve 1 at 0.4: [ 0.744  1.496]
Curve 2 at 0.4: [-0.496  1.744]

Curve 0 at 0.6: [ 0.744  0.504]
Curve 1 at 0.6: [ 0.496  1.744]
Curve 2 at 0.6: [-0.744  1.496]

Curve 0 at 0.8: [ 0.928  0.768]
Curve 1 at 0.8: [ 0.232  1.928]
Curve 2 at 0.8: [-0.928  1.232]
```



In [ ]: