r2knowle_a2q4

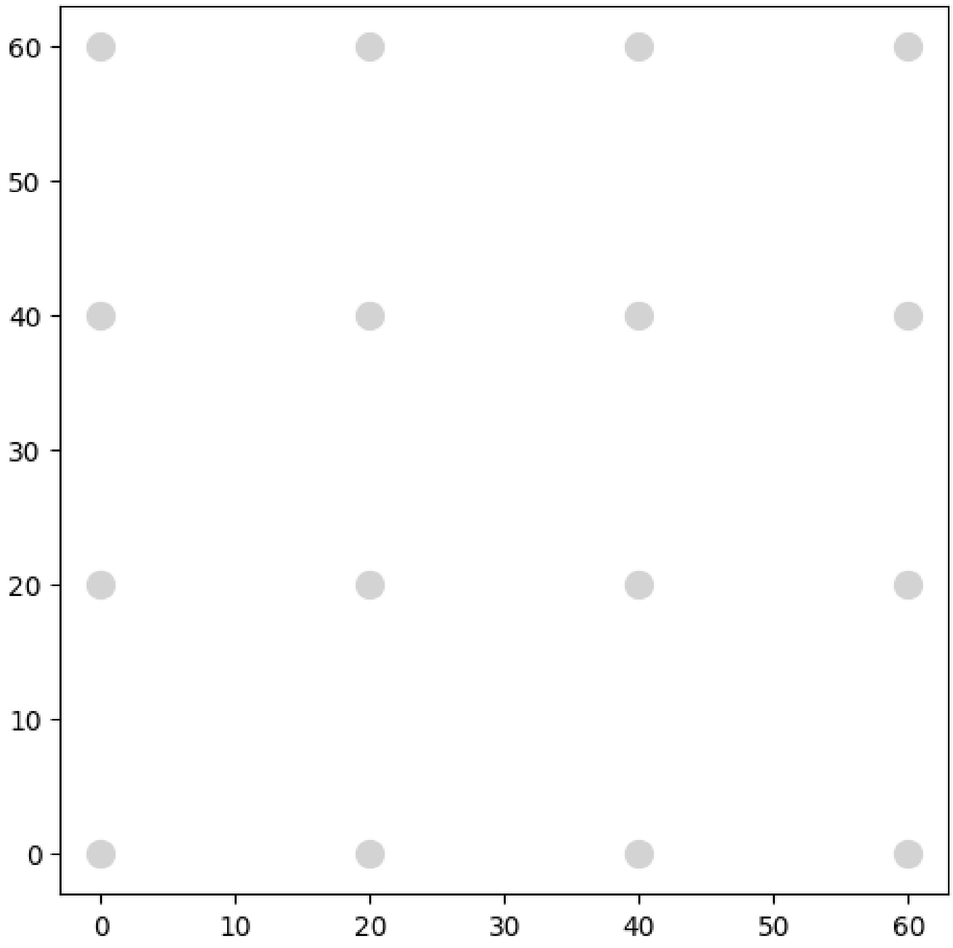
February 4, 2023

1 A2-Q4: Unlock Code

[1]:
```
import numpy as np
from scipy.interpolate import make_interp_spline
import matplotlib.pyplot as plt
```

[2]:
```
# Display grid of 16 circles
def DrawGrid():
    plt.figure(figsize=(6,6))
    [gx, gy] = np.meshgrid([0, 20, 40, 60], [0, 20, 40, 60])
    plt.plot(gx, gy,'o', color='lightgray', markersize=10); plt.axis('square');
```

[3]:
```
DrawGrid()
```



1.1 (a) Fit Points with a Spline

[7]:
```
# === YOUR CODE HERE ===
time = [0, 878, 1424, 1969, 2737]
x_points = [1, 60, 32, 29, 44]
y_points = [59, 24, 22, 42, 62]

# use spline from Q1
def MySpline(x, y):
    '''
     S = MySpline(x, y)

     Input:
       x and y are arrays (or lists) of corresponding x- and y-values,
       specifying the points in the x-y plane.  The x-values
       must be in increasing order.

     Output:
       S is a function that takes x or an array (or list) of x-values
         It evaluates the cubic spline and returns the interpolated value.

     Implementation:

       Hence...
         a[0] = a_0          b[0] = b_1        c[0] = c_1
         a[1] = a_1          b[1] = b_2        c[1] = c_2
              :                   :                 :
         a[n-2] = a_(n-2)    b[n-2] = b_(n-1)  c[n-2] = c_(n-1)
         a[n-1] = a_(n-1)

       The polynomial piece is evaluated at xx using

         p_i(xx) = a[i]*(x[i+1]-xx)**3/(6*hi) + a[i+1]*(xx-x[i])**3/(6*hi) +
                   b[i]*(x[i+1]-xx) + c[i]*(xx-x[i])

       where hk = x[k+1] - x[k] for k = 0, ... , n-2
    '''
    n = len(x)
    h = np.zeros(n-1)
    b = np.zeros(n-1)
    c = np.zeros(n-1)
    a = np.zeros(n)

    M = np.zeros((n,n))
    r = np.zeros(n)



    # === YOUR CODE HERE ===

    # Determine h first:

    for i in range(0, n-1):
        h[i] = x[i+1] - x[i]


    # Now we need to determine a, starting with the first matrix

    M[0][0] = h[0]/3
    M[0][1] = h[0]/6

    for i in range(1, n-1):
        M[i][i-1] = h[i-1]/6
        M[i][i] = (h[i-1] + h[i])/3
        M[i][i+1] = h[i]/6

    M[n-1][n-2] = h[0]/6
    M[n-1][n-1] = h[0]/3

    # Now we need to determine the second matrix:
    for i in range (1, n-1):
        r[i] = (y[i+1] - y[i])/h[i] - (y[i] - y[i-1])/h[i-1]

    # Lets now solve the array for the values of a
    a = np.linalg.solve(M, r)
    a[0] = 0           # make sure ending points are zero for natural BCs
    a[n-1] = 0         # make sure ending points are zero for natural BCs



    # Now we can determine both b and c
    for i in range(0, n-1):
        b[i] = y[i]/h[i] - a[i]*h[i]/6

    for i in range(0, n-1):
        c[i] = y[i+1]/h[i] - a[i+1]*h[i]/6



    #=======================================
    #
    # This is the function that gets returned.
    # It evaluates the cubic spline at xvals.
    #
    def spline(xvals, x=x, a=a, b=b, c=c):
        '''
         S = spline(xvals)

         Evaluates the cubic spline at xvals.

         Inputs:
          xvals can be list-like, or a scalar (**must be in ascending order**)

         Output:
          S is a list of values with the same number of elements as x
        '''
        # Turn non-list-like input into list-like
        if type(xvals) not in (list, np.ndarray,):
            xvals = [xvals]

        S = []  # The return list of values
        #
        k = 0   # this is the current polynomial piece
        hk = x[k+1] - x[k]

        print(x)
        for xx in xvals:

            # If the next x-value is not on the current piece...
            if xx>x[k+1]:
                # ... Go to next piece
                k += 1
                hk = x[k+1] - x[k]

            S_of_x = a[k]*(x[k+1]-xx)**3/(6*hk) + a[k+1]*(xx-x[k])**3/(6*hk) + b[k]*(x[k+1]-xx) + c[k]*(xx-x[k])

            S.append(S_of_x)

        return S
    #=======================================

    return spline

xcalc = MySpline(time, x_points)
ycalc = MySpline(time, y_points)
```

1.2 (b) Plot the Spline

[11]:

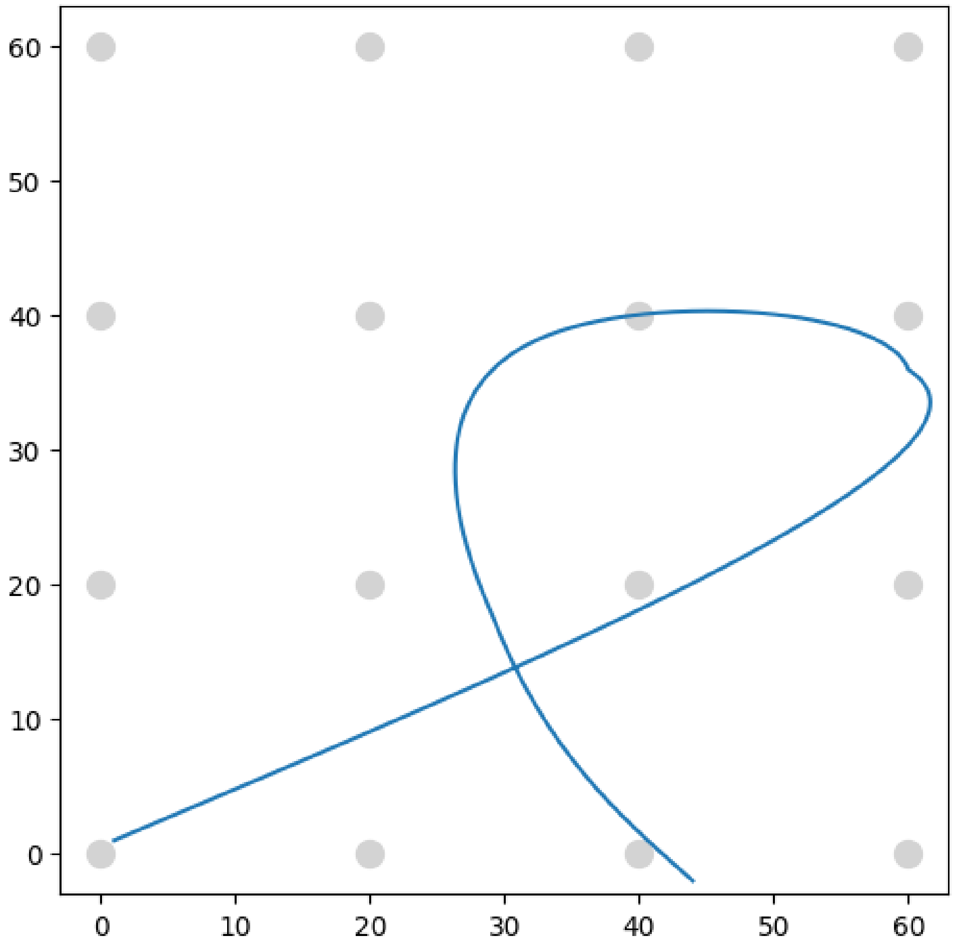
```
# === YOUR CODE HERE ===
DrawGrid()

#bottom left is (0,0) so we need to make the y values negative
points = np.linspace(time[0], time[-1], 2000)
plt.plot(xcalc(points), list(map(lambda vals: 60-vals, ycalc(points))))
```

```
[0, 878, 1424, 1969, 2737]
[0, 878, 1424, 1969, 2737]
```

[11]:

```
[<matplotlib.lines.Line2D at 0x1ffabf79a80>]
```



1.3 (c) Unlock Pattern

=== YOUR ANSWER HERE ===

[ ]:
```
The order of points is:
    1, 7, 12, 11, 3
So the overall pattern is 1-7-12-11-3
```