HOMEWORK: REPRESENTATION AND MANIPULATION OF CURVES

Name: Nguyễn Chế Thanh

Student ID:2170103

Professor: Bành Quốc Nguyên

I) Problem statement

A 2.5D workpiece profile was measured by a contact probe with the radius of 1.0 mm on a coordinate measuring machine (CMM). The probe center coordinates of measured data points on x-z plane are as follows:

(0.500, 2.150)

(4.325, 2.225)

(8.545, 3.192)

(13.180, 4.533)

(17.830, 6.291)

(21.150, 7.146)

(24.480, 7.575)

(28.160, 7.263)

(32.395, 6.147)

(36.150, 4.658)

(42.595, 2.572)

(49.000, 2.160)

(1) Calculate and plot the coordinates of the data points on the workpiece after the probe radius compensation by using composite Ferguson spline method.

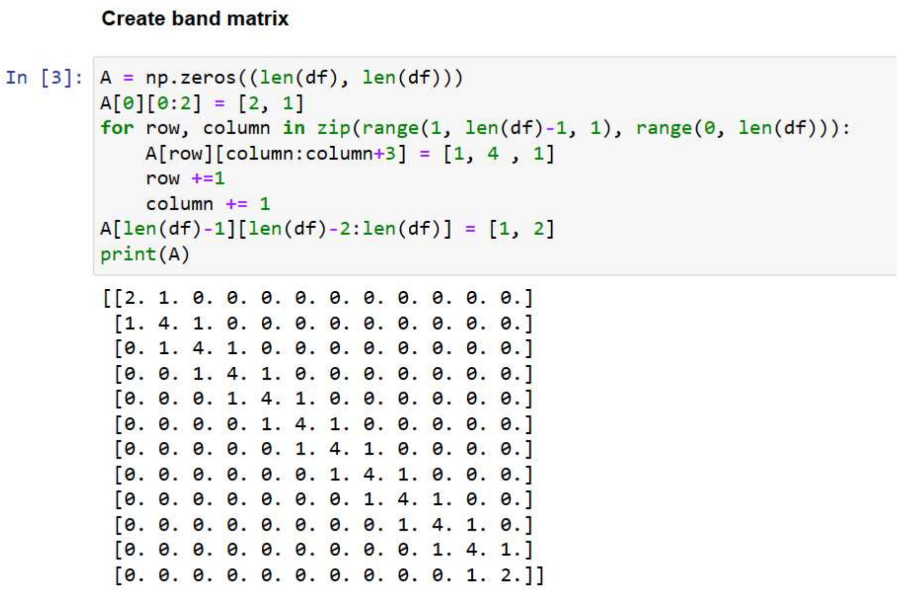
(2) Calculate and plot the control vertices of the uniform composite B-spline curve passing through the data points on the workpiece surface.

(3) Using the composite Ferguson spline method and B-spline curve method, to interpolate and compare the data points on the workpiece surface at x=20.000 and x=35.000, respectively.

II) Solution

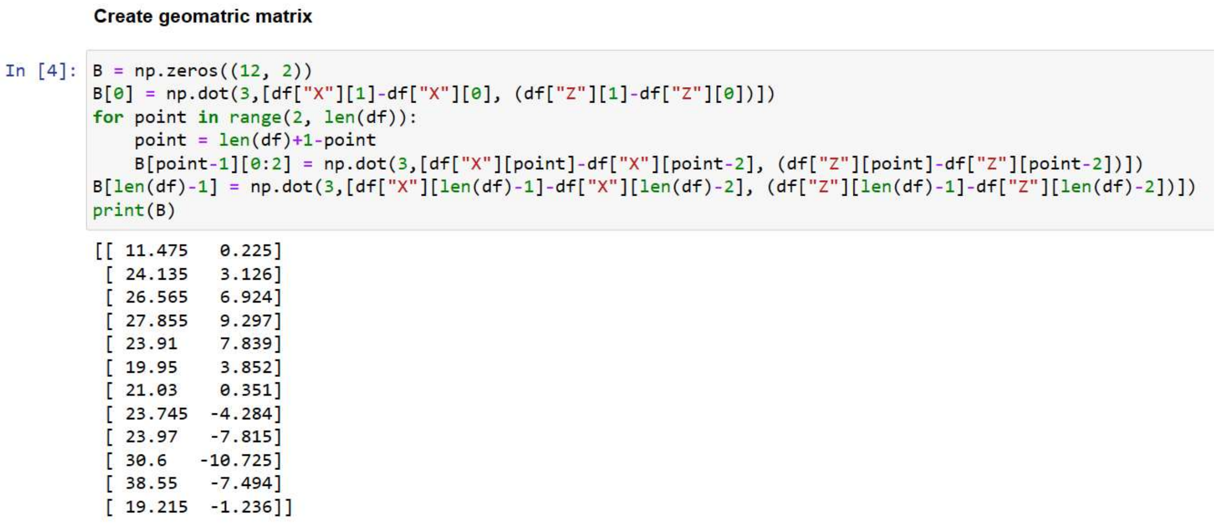
Problem (1):

- General form for a composite Ferguson spline passing through (n) data points:



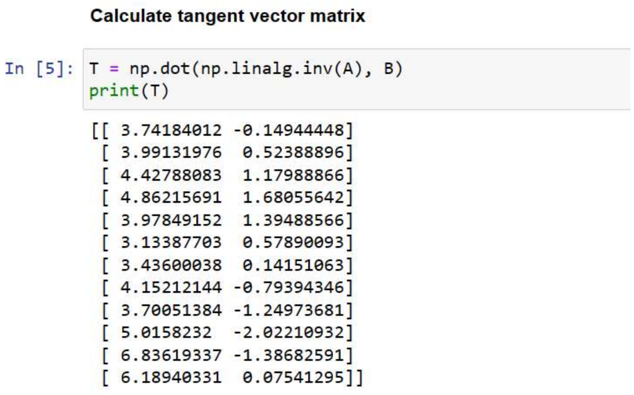
Create band matrix

```
In [3]: A = np.zeros((len(df), len(df)))
        A[0][0:2] = [2, 1]
        for row, column in zip(range(1, len(df)-1, 1), range(0, len(df))):
            A[row][column:column+3] = [1, 4 , 1]
            row +=1
            column += 1
        A[len(df)-1][len(df)-2:len(df)] = [1, 2]
        print(A)

        [[2. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
         [1. 4. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
         [0. 1. 4. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
         [0. 0. 1. 4. 1. 0. 0. 0. 0. 0. 0. 0.]
         [0. 0. 0. 1. 4. 1. 0. 0. 0. 0. 0. 0.]
         [0. 0. 0. 0. 1. 4. 1. 0. 0. 0. 0. 0.]
         [0. 0. 0. 0. 0. 1. 4. 1. 0. 0. 0. 0.]
         [0. 0. 0. 0. 0. 0. 1. 4. 1. 0. 0. 0.]
         [0. 0. 0. 0. 0. 0. 0. 1. 4. 1. 0. 0.]
         [0. 0. 0. 0. 0. 0. 0. 0. 1. 4. 1. 0.]
         [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 4. 1.]
         [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 2.]]
```



Create geomatric matrix

```
In [4]: B = np.zeros((12, 2))
        B[0] = np.dot(3,[df["X"][1]-df["X"][0], (df["Z"][1]-df["Z"][0])])
        for point in range(2, len(df)):
            point = len(df)+1-point
            B[point-1][0:2] = np.dot(3,[df["X"][point]-df["X"][point-2], (df["Z"][point]-df["Z"][point-2])])
        B[len(df)-1] = np.dot(3,[df["X"][len(df)-1]-df["X"][len(df)-2], (df["Z"][len(df)-1]-df["Z"][len(df)-2])])
        print(B)

        [[ 11.475   0.225]
         [ 24.135   3.126]
         [ 26.565   6.924]
         [ 27.855   9.297]
         [ 23.91    7.839]
         [ 19.95    3.852]
         [ 21.03    0.351]
         [ 23.745  -4.284]
         [ 23.97   -7.815]
         [ 30.6   -10.725]
         [ 38.55   -7.494]
         [ 19.215  -1.236]]
```

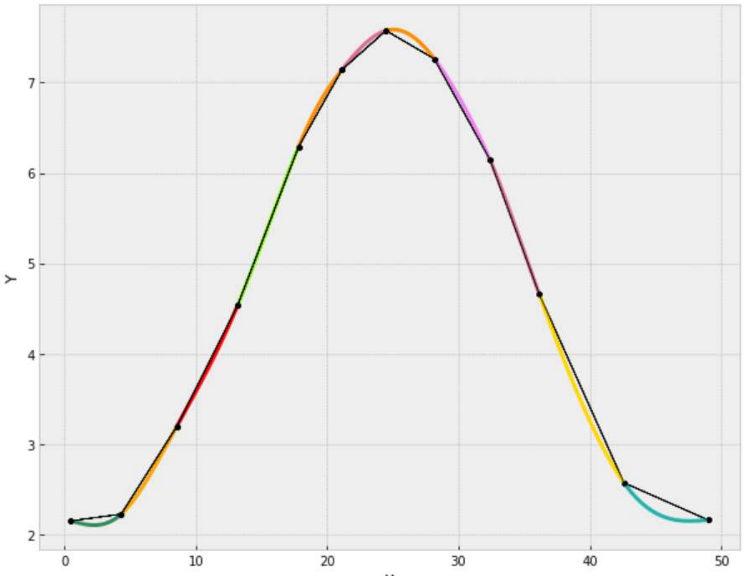


Calculate tangent vector matrix

```
In [5]: T = np.dot(np.linalg.inv(A), B)
        print(T)

        [[ 3.74184012 -0.14944448]
         [ 3.99131976  0.52388896]
         [ 4.42788083  1.17988866]
         [ 4.86215691  1.68055642]
         [ 3.97849152  1.39488566]
         [ 3.13387703  0.57890093]
         [ 3.43600038  0.14151063]
         [ 4.15212144 -0.79394346]
         [ 3.70051384 -1.24973681]
         [ 5.0158232  -2.02210932]
         [ 6.83619337 -1.38682591]
         [ 6.18940331  0.07541295]]
```

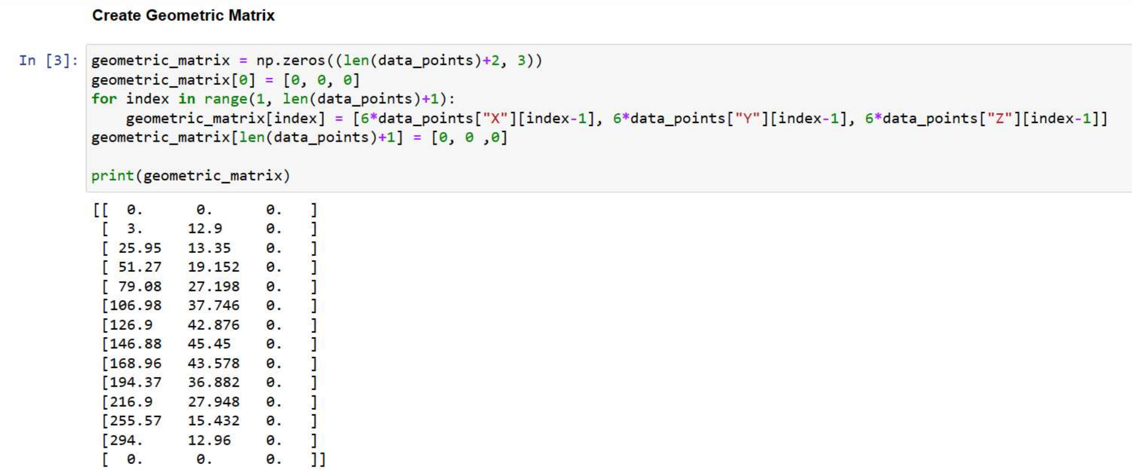
Plot Curve

In [7]:
```python
points_num = 41
fig = plt.figure(figsize=(10, 8)).add_subplot()
for i in range(0, len(df)-1, 1):
    u = 0
    pointsX = []
    pointsY = []
    boundary_condition = np.array([[df["X"][i], df["Z"][i]],[df["X"][i+1], df["Z"][i+1]],T[i],T[i+1]])
    for point_num in range(0, points_num):
        f1_u = 1 - 3*(u**2) + 2*(u**3)
        f2_u = 3*(u**2) - 2*(u**3)
        f3_u = u - 2*(u**2) + (u**3)
        f4_u = -(u**2) + u**3
        blending_function = np.array([f1_u, f2_u, f3_u, f4_u])
        u = u + 1/(points_num-1)
        pointsX.append(np.dot(blending_function, boundary_condition)[0])
        pointsY.append(np.dot(blending_function, boundary_condition)[1])
    plt.plot(pointsX, pointsY, color=random.choice(color), lw=3)
    plt.plot(df["X"], df["Z"], marker=".", color="black", lw=0.5)
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```



Problem (2):

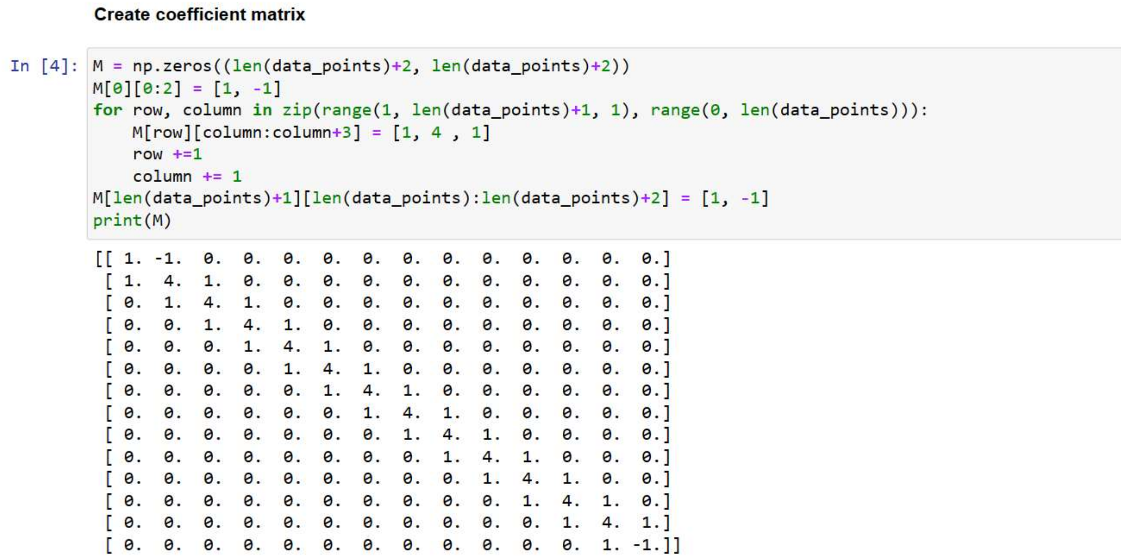
-Find control vertices of the uniform composite B-spline curve passing through (n+2) data points on the workpiece surface:



Create Geometric Matrix

In [3]:
```
geometric_matrix = np.zeros((len(data_points)+2, 3))
geometric_matrix[0] = [0, 0, 0]
for index in range(1, len(data_points)+1):
    geometric_matrix[index] = [6*data_points["X"][index-1], 6*data_points["Y"][index-1], 6*data_points["Z"][index-1]]
geometric_matrix[len(data_points)+1] = [0, 0 ,0]

print(geometric_matrix)
```

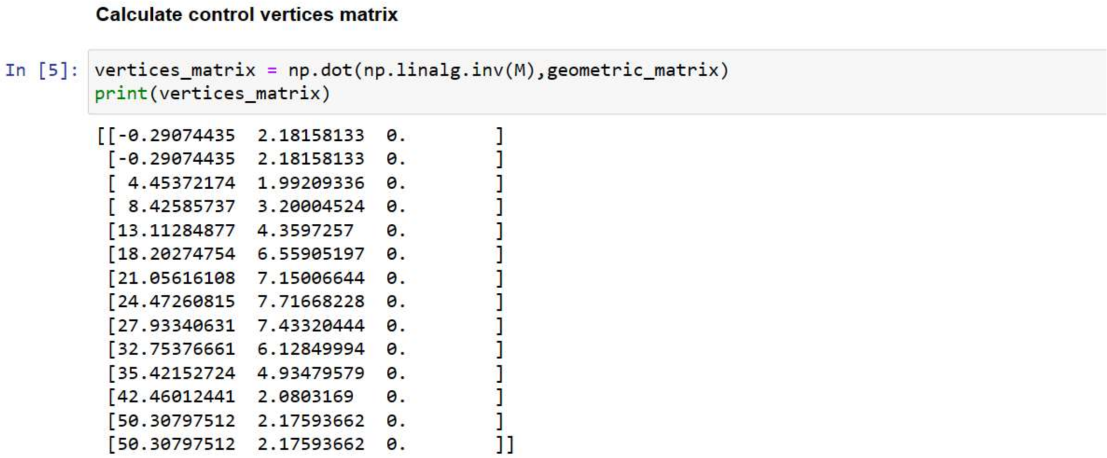
```
[[  0.       0.       0.    ]
 [  3.      12.9      0.    ]
 [ 25.95    13.35     0.    ]
 [ 51.27    19.152    0.    ]
 [ 79.08    27.198    0.    ]
 [106.98    37.746    0.    ]
 [126.9     42.876    0.    ]
 [146.88    45.45     0.    ]
 [168.96    43.578    0.    ]
 [194.37    36.882    0.    ]
 [216.9     27.948    0.    ]
 [255.57    15.432    0.    ]
 [294.      12.96     0.    ]
 [  0.       0.       0.    ]]
```



Create coefficient matrix

In [4]:
```
M = np.zeros((len(data_points)+2, len(data_points)+2))
M[0][0:2] = [1, -1]
for row, column in zip(range(1, len(data_points)+1, 1), range(0, len(data_points))):
    M[row][column:column+3] = [1, 4 , 1]
    row +=1
    column += 1
M[len(data_points)+1][len(data_points):len(data_points)+2] = [1, -1]
print(M)
```

```
[[ 1. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  4.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  4.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  4.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  4.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.  4.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  1.  4.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  1.  4.  1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  1.  4.  1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  4.  1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  4.  1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  4.  1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  4.  1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1. -1.]]
```



Calculate control vertices matrix

In [5]:
```
vertices_matrix = np.dot(np.linalg.inv(M),geometric_matrix)
print(vertices_matrix)
```

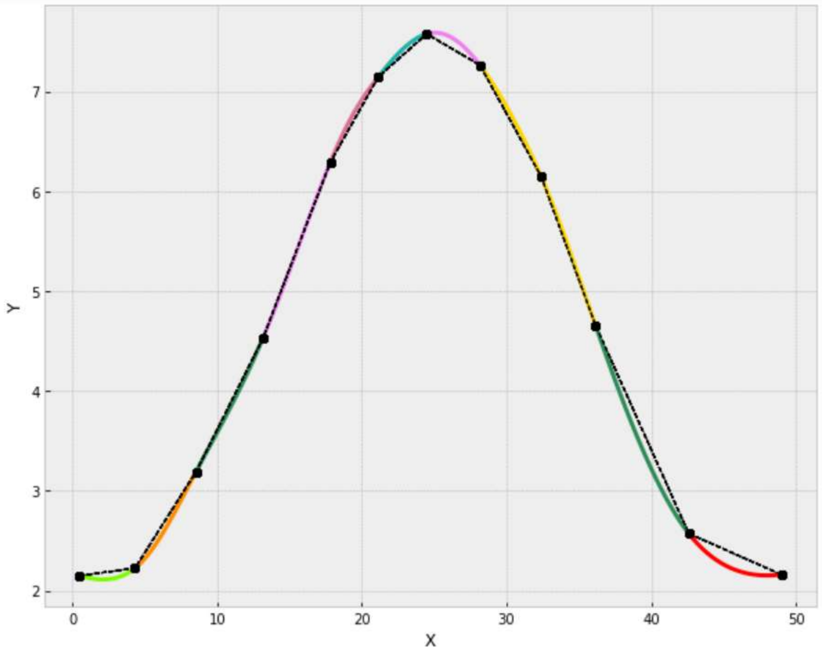
```
[[-0.29074435  2.18158133  0.        ]
 [-0.29074435  2.18158133  0.        ]
 [ 4.45372174  1.99209336  0.        ]
 [ 8.42585737  3.20004524  0.        ]
 [13.11284877  4.3597257   0.        ]
 [18.20274754  6.55905197  0.        ]
 [21.05616108  7.15006644  0.        ]
 [24.47260815  7.71668228  0.        ]
 [27.93340631  7.43320444  0.        ]
 [32.75376661  6.12849994  0.        ]
 [35.42152724  4.93479579  0.        ]
 [42.46012441  2.0803169   0.        ]
 [50.30797512  2.17593662  0.        ]
 [50.30797512  2.17593662  0.        ]]
```

Create constant matrix

```
In [6]: con_matrix = [[-1, 3, -3, 1],
                      [3, -6, 3, 0],
                      [-3, 0, 3, 0],
                      [1, 4, 1, 0]]
        con_matrix = np.array(con_matrix)
```

Plot the curve

```
In [7]: points_num = 41
        pointsX = []
        pointsY = []
        plt.style.use("bmh")
        fig = plt.figure(figsize=(10, 8)).add_subplot()
        for i in range(0, len(data_points) + 2 - 3, 1):
            u = 0
            control_points = vertices_matrix[i:i+4]
        #     print(control_points)
            for point_num in range(0, points_num):
                f1_u = (1/6)*u**3
                f2_u = (1/6)*u**2
                f3_u = (1/6)*u
                f4_u = 1/6
                blending_function = np.array([f1_u, f2_u, f3_u, f4_u])
                u = u + 1 / (points_num - 1)
                pointsX.append(np.dot(np.dot(blending_function, con_matrix), control_points)[0])
                pointsY.append(np.dot(np.dot(blending_function, con_matrix), control_points)[1])
            plt.plot(pointsX, pointsY, color=random.choice(color), lw=3)
            plt.plot(data_points["X"], data_points["Y"], marker="o", color="black", ls="--", lw=1)
            pointsX = []
            pointsY = []
        plt.xlabel("X")
        plt.ylabel("Y")
        plt.show()
```



Problem (3):

Ferguson spline equations			
X =	20	20.00197	5.926234
X =	35	35.00632	3.584316
B-spline curve equations			
X =	20	20.00056	5.948948
X =	35	35.02775	3.64145