```
import numpy as np
from sympy import sympify


class QuadraticSpline():

    def expandParams(self,X, Y):
        n = len(X)
        new_X = np.zeros(n)
        new_Y = np.zeros(n)

        for i in range(n):
            new_X[i] = X[i]
            new_Y[i] = Y[i]

        points = np.array((new_X, new_Y)).T
        return points

    def compute(self, params):
        X = params["X"]
        Y = params["Y"]

        points = self.expandParams(X, Y)

        n = len(points) - 1
        points = np.array(points)
        matrix = np.zeros((n*3, n*3))
        indVector = np.zeros(n*3)

        j = 0
        k = 0
        for i in range(0, n*2, 2):
            matrix[i, j+0] = points[k, 0] ** 2
            matrix[i, j+1] = points[k, 0]
            matrix[i, j+2] = 1

            matrix[i+1, j+0] = points[k+1, 0] ** 2
            matrix[i+1, j+1] = points[k+1, 0]
            matrix[i+1, j+2] = 1

            j += 3
            k += 1

        j = 1
        k = 0
        for i in range(n*2, n*3-1):
            matrix[i][k + 0] = 2 * points[j, 0]
            matrix[i][k + 1] = 1

            matrix[i][k + 2+1] = - 2 * points[j, 0]
            matrix[i][k + 3+1] = - 1
            j += 1
            k += 3

        matrix[n*3-1, 0] = 1

        indVector[0] = points[0, 1]
        j = 1
        for i in range(1, n):
            indVector[j] = points[i, 1]
            indVector[j+1] = points[i, 1]
            j += 2


        solution = np.linalg.solve(matrix, indVector)
        function =  self.generateEquation(solution, points)
        return function


    def generateEquation(self, coefficients, points):
        segmentFunction = []
        coefficients = np.round(coefficients, 2)
        n = len(points) - 1

        for i in range(0, n*3, 3):
            function = "{a}x^2 + {b}x + {c}".format(
                a=coefficients[i],
                b=coefficients[i+1],
                c=coefficients[i+2],
            )
            segmentFunction.append([function, "{x0} <= x <= {x1}".format(
                    x0=points[i//3, 0],
                    x1=points[i//3+1, 0]
                )])


        return segmentFunction

points = {"X":[0,1,2,3],"Y":[0,1,1,0]}
x = QuadraticSpline()
print(x.compute(points))
```