Interpolação polinomial por partes via splines cúbicas

Vinicius Ferraço Arruda

Universidade Federal do Espírito Santo - UFES
Departamento de Informática
CEP 29060-270, Vitória, ES
viniciusferracoarruda@gmail.com

Resumo Este trabalho apresenta de forma simples e direta o método da interpolação polinomial por partes via splines cúbicas e a implementação do algoritmo em linguagem de programação *python* junto a quatro problemas com os resultados da solução obtidos a partir desta implementação.

1 Introdução

A interpolação é um método muito utilizado na engenharia e ciência em geral, onde se obtém uma amostragem de dados pontuais de um determinado experimento ou fenômeno e é desejável saber quais valores seriam mapeados por tal comportamento no intervalo dos dados coletados. Outra utilização comum da interpolação é a reconstituição aproximada de funções, bastando conhecer apenar alguns pontos da função e sua imagem.

Este trabalho tem como objetivo apresentar um método particular de interpolação, apresentando o método numérico da seção 2, resultados aplicando o método numérico a alguns problemas na seção 3 e na seção 4 o código desenvolvido a partir deste método.

2 Método numérico

Dado os pontos $[x_i, y_i]$ para $i = 0, 1, \ldots, n$ sendo $y = f(x)$, o que gera $n + 1$ pontos e n intervalos entre eles, o método da interpolação polinomial por partes via splines cúbicas é encontrar n polinômios cúbicos, onde o i-ésimo polinômio passa pelos pontos x_i e x_{i+1}, e é escrito por:

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \quad x \in [x_i, x_{i+1}] \tag{1}$$

e todos os n polinômios são denotados por $S(x)$.

Uma vez que existem n intervalos e quatro coeficientes para cada um, é exigido um total de $4n$ parâmetros para definir $S(x)$. Logo, é necessário encontrar $4n$ condições independentes.

Tem-se duas condições para cada intervalo a partir da necessidade que o polinômio cúbico corresponda aos pontos nas extremidades do intervalo:

$$\begin{aligned} S_i(x_i) &= y_i \\ S_i(x_{i+1}) &= y_{i+1} \end{aligned} \tag{2}$$

Observe que essas condições resultam em uma função contínua por partes. Ainda é preciso encontrar $2n$ condições. Uma vez que é desejado encontrar uma interpolação o mais suave possível, é exigido que a primeira e a segunda derivadas também sejam contínuas:

$$S'_{i-1}(x_i) = S'_i(x_i) \tag{3}$$

$$S''_{i-1}(x_i) = S''_i(x_i) \tag{4}$$

Essas condições se aplicam para $i = 1, 2, \ldots, n-1$, resultando em $2(n-1)$ restrições, restando mais duas condições a serem definidas para gerar a spline. Algumas escolhas podem ser feitas, e neste trabalho será apresentado a spline cúbica natural, que acrescenta as duas condições restantes:

$$\begin{aligned} S''_0(x_0) &= 0 \\ S''_{n-1}(x_n) &= 0 \end{aligned} \tag{5}$$

Com as $4n$ condições, resta apenas calcular o sistema de equações para encontrar os $4n$ coeficientes. Estas condições podem ser reduzidas, através de manipulação algébrica, a um sistema tridiagonal com apenas $n-1$ equações.

2.1 Redução algébrica

Para $x = x_i$ na Equação 1 tem-se:

$$s_i(x_i) = d_i \tag{6}$$

e de Equação 2 e Equação 6:

$$d_i = y_i, \quad i = 0, 1, 2, \ldots, n-1 \tag{7}$$

Para $x = x_{i+1}$ na Equação 1 e com a Equação 2, tem-se:

$$s_i(x_{i+1}) = s_{i+1}(x_{i+1}) = y_{i+1} \tag{8}$$

e

$$a_i(x_{i+1} - x_i)^3 + b_i(x_{i+1} - x_i)^2 + c_i(x_{i+1} - x_i) + d_i = y_{i+1} \tag{9}$$

Definindo:

$$h_i = x_{i+1} - x_i \tag{10}$$

e substituindo com a Equação 7:

$$a_i h_i^3 + b_i h_i^2 + c_i h_i + y_i = y_{i+1} \tag{11}$$

Tem-se também que as derivadas da Equação 1 são:

$$s_i'(x) = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i \tag{12}$$

$$s_i''(x) = 6a_i(x - x_i) + 2b_i \tag{13}$$

Para $x = x_i$ na Equação 13:

$$b_i = \frac{s_i''(x_i)}{2}, \quad i = 0, 1, 2, \ldots, n-1 \tag{14}$$

Para $x = x_{i+1}$ na Equação 13:

$$s_i''(x_{i+1}) = 6a_i(x_{i+1} - x_i) + 2b_i \tag{15}$$

Em vista da Equação 4 e substituindo com as Equações 10 e 14:

$$s_{i+1}''(x_{i+1}) = 6a_i h_i + 2\frac{s_i''(x_i)}{2}$$

$$a_i = \frac{s_{i+1}''(x_{i+1}) - s_i''(x_i)}{6h_i}, \quad i = 0, 1, 2, \ldots, n-1 \tag{16}$$

Substituindo as Equações 7, 14 e 16 na Equação 11:

$$\frac{s_{i+1}''(x_{i+1}) - s_i''(x_i)}{6h_i}h_i^3 + \frac{s_i''(x_i)}{2}h_i^2 + c_i h_i + y_i = y_{i+1}$$

obtém-se:

$$c_i = \mathit{\Delta} y_i - \frac{s_{i+1}''(x_{i+1}) + 2s_i''(x_i)}{6}h_i, \quad i = 0, 1, 2, \ldots, n-1 \tag{17}$$

sendo o operador de diferença dividida $\mathit{\Delta} y_i$ definido por:

$$\mathit{\Delta} y_i = \frac{y_{i+1} - y_i}{h_i} \tag{18}$$

Substituindo a Equação 12 na Equação 3:

$$3a_{i-1}(x_i - x_{i-1})^2 + 2b_{i-1}(x_i - x_{i-1}) + c_{i-1} = 3a_i(x_i - x_i)^2 + 2b_i(x_i - x_i) + c_i$$

e substituindo 10, 14, 16 e 17:

$$\begin{aligned} &3\frac{s_i''(x_i) - s_{i-1}''(x_{i-1})}{6h_{i-1}}h_{i-1}^2 + 2\frac{s_{i-1}''(x_{i-1})}{2}h_{i-1} + \frac{y_i - y_{i-1}}{h_{i-1}} \\ &- \frac{s_i''(x_i) + 2s_{i-1}''(x_{i-1})}{6}h_{i-1} = \frac{y_{i+1} - y_i}{h_i} - \frac{s_{i+1}''(x_{i+1}) + 2s_i''(x_i)}{6}h_i \end{aligned}$$

e simplificando, obtém-se a i-ésima equação para $i = 1, 2, 3, \ldots, n-1$:

$$h_{i-1}s_{i-1}''(x_{i-1}) + 2(h_{i-1} + h_i)s_i''(x_i) + h_i s_{i+1}''(x_{i+1}) = 6(\mathit{\Delta} y_i - \mathit{\Delta} y_{i-1}) \tag{19}$$

que é um sistema linear subdeterminado, ou seja, quando o sistema possui menos equações que incógnitas, com $n-1$ equações e $n+1$ incógnitas $s_i''(x_i)$ com $i = 0, 1, 2, \ldots, n$.

O sistema linear descrito pela Equação 19 pode ser escrito na forma matricial a seguir:

$$\begin{bmatrix} h_0 & 2(h_0+h_1) & h_1 & & & & \\ & h_1 & 2(h_1+h_2) & h_2 & & & \\ & & h_2 & 2(h_2+h_3) & h_3 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \end{bmatrix} \times \begin{bmatrix} s_0''(x_0) \\ s_1''(x_1) \\ s_2''(x_2) \\ \vdots \\ s_{n-1}''(x_{n-1}) \\ s_n''(x_n) \end{bmatrix} = 6 \begin{bmatrix} \Delta y_1 - \Delta y_0 \\ \Delta y_2 - \Delta y_1 \\ \Delta y_3 - \Delta y_2 \\ \vdots \\ \Delta y_{n-1} - \Delta y_{n-2} \end{bmatrix} \tag{20}$$

A saber das Equações 5 e 4, é possível eliminar duas incógnitas de modo a obter um sistema linear tridiagonal simétrico de ordem $n-1$:

$$\begin{bmatrix} 2(h_0+h_1) & h_1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & h_2 & 2(h_2+h_3) & h_3 & \\ & & \ddots & \ddots & \ddots \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{bmatrix} \times \begin{bmatrix} s_1''(x_1) \\ s_2''(x_2) \\ s_3''(x_3) \\ \vdots \\ s_{n-1}''(x_{n-1}) \end{bmatrix} = 6 \begin{bmatrix} \Delta y_1 - \Delta y_0 \\ \Delta y_2 - \Delta y_1 \\ \Delta y_3 - \Delta y_2 \\ \vdots \\ \Delta y_{n-1} - \Delta y_{n-2} \end{bmatrix} \tag{21}$$

cuja solução fornece as derivadas $s_i''(x_i)$, $i = 1, 2, 3, \ldots, n-1$, restando apenas encontrar os coeficientes a_i, b_i, c_i e d_i através das Equações 16, 14, 17 e 7, respectivamente.

3 Resultados

Para os resultados, foram utilizados quatro bases de dados distintas, sendo uma inserida via teclado e outras três via arquivo.

3.1 Problema inicial

Obter as splines interpoladoras dos 6 pontos da Tabela 1, inseridos via teclado, e calcular o valor de $S(0.7)$.

i	x_i	y_i
0	0.0	0.5
1	0.4	1.5
2	1.0	1.0
3	1.5	1.1
4	1.7	0.9
5	2.0	0.5

Tabela 1: Pontos (x, y) do problema inicial

O gráfico das splines gerado por 50 pontos é dado pela Figura 1 e a solução $S(0.7) = 1.34890917827967338205$.

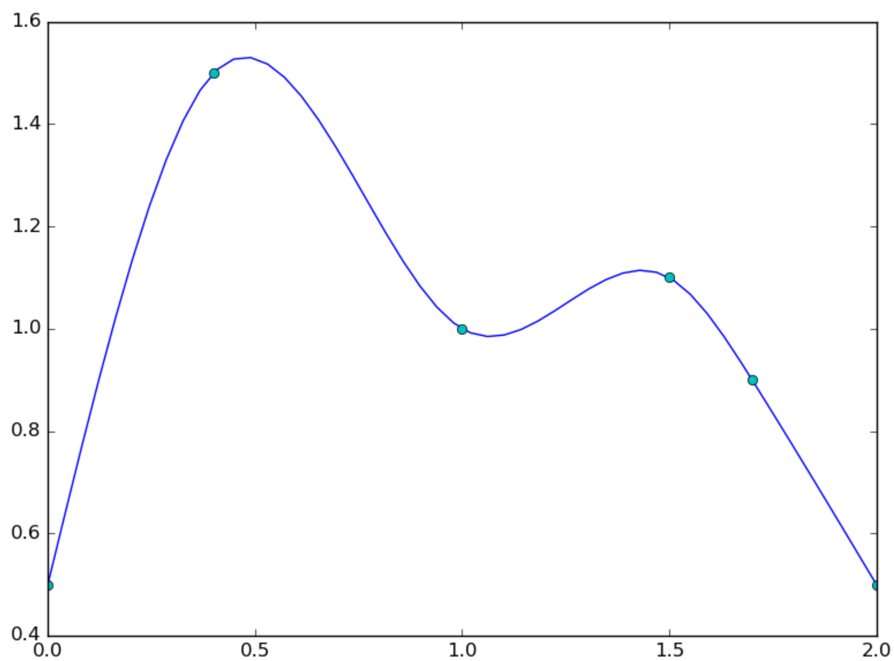

Figura 1: Splines do problema inicial

3.2 Problema arquivo 1

Obter as splines interpoladoras dos 9 pontos da Tabela 2, inseridos via arquivo *dados1.txt*, e calcular o valor de $S(5.7)$.

i	x_i	y_i
0	0.0	0.0
1	1.25	0.94898
2	2.5	0.59847
3	3.75	−0.57156
4	5.0	−0.95892
5	6.25	−0.03318
6	7.5	0.938
7	8.75	0.62472
8	10.0	−0.54402

Tabela 2: Pontos (x, y) do problema do arquivo *dados1.txt*

O gráfico das splines gerado por 50 pontos é dado pela Figura 2 e a solução $S(5.7) = -0.545787371188453440 67$.

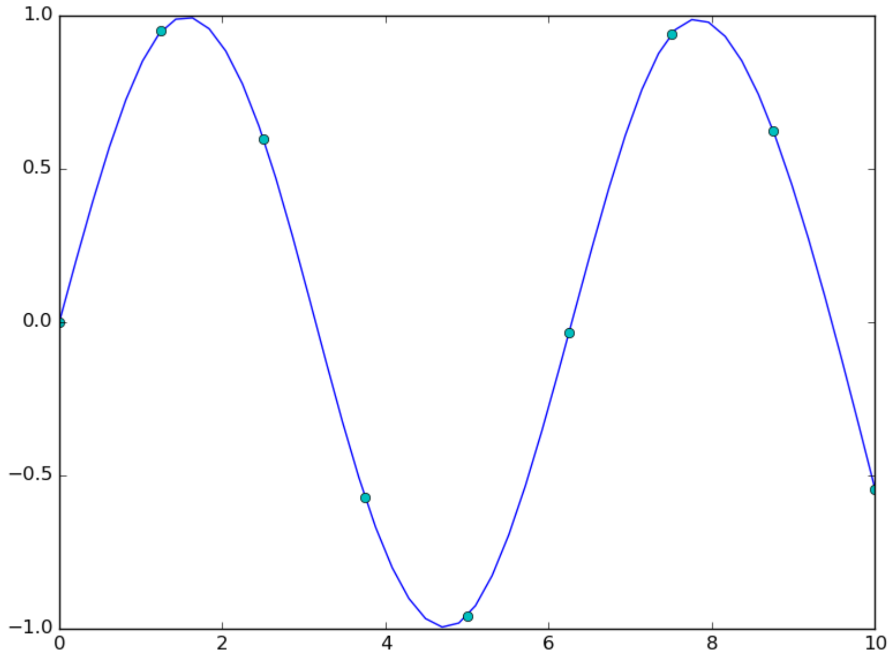

Figura 2: Splines do problema do arquivo *dados1.txt*

3.3 Problema arquivo 2

Obter as splines interpoladoras dos 6 pontos da Tabela 3, inseridos via arquivo *dados2.txt*, e calcular o valor de $S(0.7)$.

i	x_i	y_i
0	0.0	0.5
1	0.4	1.5
2	1.0	1.0
3	1.5	1.1
4	1.7	0.9
5	2.0	0.5

Tabela 3: Pontos (x, y) do problema do arquivo *dados2.txt*

O gráfico das splines gerado por 100 pontos é dado pela Figura 3 e a solução $S(0.7) = 1.34890917827967338205$.

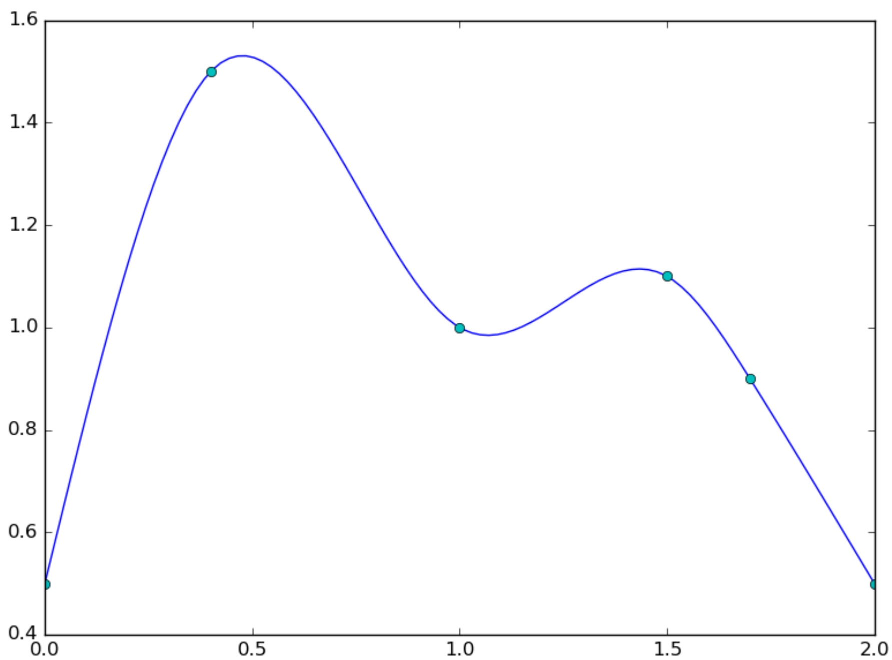

Figura 3: Splines do problema do arquivo *dados2.txt*

3.4 Problema arquivo 3

Obter as splines interpoladoras dos 21 pontos da Tabela 4, inseridos via arquivo *dados3.txt*, e calcular o valor de $S(-0.1)$.

i	x_i	y_i	i	x_i	y_i	i	x_i	y_i
0	-1.0	0.038462	7	-0.3	0.307692	14	0.4	0.2
1	-0.9	0.047059	8	-0.2	0.5	15	0.5	0.137931
2	-0.8	0.058824	9	0.1	0.8	16	0.6	0.1
3	-0.7	0.075472	10	0.0	1.0	17	0.7	0.075472
4	-0.6	0.1	11	0.1	0.8	18	0.8	0.058824
5	-0.5	0.137931	12	0.2	0.5	19	0.9	0.047059
6	-0.4	0.2	13	0.3	0.307692	20	1.0	0.038462

Tabela 4: Pontos (x, y) do problema do arquivo *dados3.txt*

O gráfico das splines gerado por 100 pontos é dado pela Figura 4 e a solução $S(-0.1) = 0.80000000000000004441$.

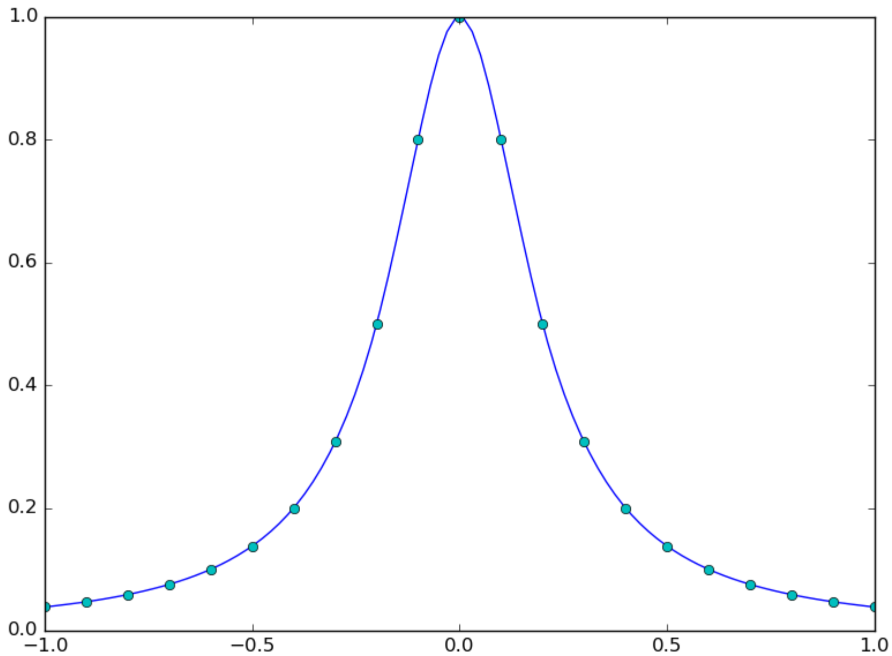

Figura 4: Splines do problema do arquivo *dados3.txt*

4 Código

Para executar o programa via terminal, execute o interpretador de *python* com o módulo principal *Main.py* como a seguir:

```
$ python Main.py
```

Para executar em algum outro interpretador *python*, apenas chame o módulo *Main.py*, e mantenha os módulos *Interface.py* e *Spline.py* no mesmo diretório.

Inicialmente, um menu de entrada será exibido como a seguir:

```
______Menu______
1 - Ler dados via teclado
2 - Ler dados via arquivo
3 - Sair
Escolha:
```

Caso a opção 2 seja escolhida, será exibido o seguinte menu:

```
______Menu______
1 - Ler arquivo dados1.txt
2 - Ler arquivo dados2.txt
3 - Ler arquivo dados3.txt
4 - Sair
Escolha:
```

Todas as entradas de dados, tanto via teclado quanto via arquivo devem seguir o seguinte modelo:

n
$x_1, x_2, \ldots, x_n$
$y_1, y_2, \ldots, y_n$
z

onde n é o número de pontos e z é um valor arbitrário no intervalo $[x_1, x_n]$ no qual, ao fim do programa, o valor de $S(z)$ será impresso na tela.

Após a inserção dos dados ou escolha do arquivo, o programa irá executar os cálculos e em seguida gerar a saída, sendo o valor de $S(z)$ exibido na tela, um arquivo chamado *grafico.txt* que contém m pontos espaçados igualmente dentro do intervalo $[x_1, x_n]$ e seus respectivos mapeamentos $S(x_i)$ e um arquivo *grafico.png* que contém o gráfico gerado a partir dos dados do arquivo *grafico.txt* com os pontos (x_i, y_i) iniciais em destaque. O valor de m ficou definido como 50 para a entrada via teclado e para o arquivo 1 e para os arquivos 2 e 3, $m = 100$.

O Apêndice A contém o código do programa em linguagem de programação *python*.

Referências

1. Filho, Frederico F.C. Algoritmos numéricos. LTC, 2007
2. Chapra, Steven C., and Raymond P. Canale. *Numerical methods for engineers*. Vol. 2. McGraw-Hill, 2012.

A Implementação em *Python* do método numérico

O programa foi dividido em três módulos:

Main.py: Módulo responsável por inicializar o programa e chamar o módulo responsável pela interface.

Interface.py: Módulo responsável por gerenciar a interface do programa com o usuário, exibindo o menu e lendo a entrada de dados do usuário via teclado ou arquivo, repassando estes dados ao módulo responsável pelo método numérico.

Spline.py: Módulo responsável pelo método numérico, com as funcionalidades de calcular e avaliar as splines interpoladoras e gerar o gráfico e o arquivo com os m pontos (x, y).

A.1 Código do arquivo Main.py

```
from Interface import Interface

def main():
  I = Interface()
  I.start()

if __name__ == "__main__":
  main()
```

A.2 Código do arquivo Interface.py

```
from Spline import Spline

class Interface:

  def start(self):
    while True:
      choice = self._getChoice(self._getMenu())
      (n,X,Y,z,m) = self._getDataByChoice(choice)
      s = Spline(X,Y)
      print '%.20f' % s.getY(z)
      s.savePlot('grafico',m)

  def _getMenu(self):
    return "___________Menu____________\n" + \
           "1 - Ler dados via teclado\n" + \
           "2 - Ler dados via arquivo\n" + \
           "3 - Sair\n" + \
           "Escolha: "

  def _getChoice(self, message):
```

```
        return int(raw_input(message))

    def _getDataByChoice(self, choice):
        if choice == 1:   return self._getDataByKeyboard()
        elif choice == 2: return self._getDataByFile()
        elif choice == 3: exit(0)

    def _getDataByKeyboard(self):
        n = int(raw_input())
        X = map(float, raw_input().split())
        Y = map(float, raw_input().split())
        z = float(raw_input())
        m = 50
        return (n, X, Y, z, m)

    def _getFileMenu(self):
        return "____________Menu____________\n" + \
               "1 - Ler arquivo dados1.txt\n" + \
               "2 - Ler arquivo dados2.txt\n" + \
               "3 - Ler arquivo dados3.txt\n" + \
               "4 - Sair\n" + \
               "Escolha: "

    def _getDataByFileChoice(self, choice):
        if choice == 1:
            return self._getDataFromFile("dados1.txt") + (50,)
        elif choice == 2:
            return self._getDataFromFile("dados2.txt") + (100,)
        elif choice == 3:
            return self._getDataFromFile("dados3.txt") + (100,)
        elif choice == 4:
            exit(0)

    def _getDataFromFile(self, filename):
        lines = [line.rstrip('\n') for line in open(filename)]
        n = int(lines[0])
        X = map(float, lines[1].split())
        Y = map(float, lines[2].split())
        z = float(lines[3])
        return (n, X, Y, z)

    def _getDataByFile(self):
        choice = self._getChoice(self._getFileMenu())
        return self._getDataByFileChoice(choice)
```

A.3 Código do arquivo Spline.py

```
from __future__ import division
import matplotlib.pyplot as plt
```

```

class Spline:

  def __init__(self,x,y):
    self.x = [float(xx) for xx in x]
    self.y = [float(yy) for yy in y]
    self.c = self._getCoeficients()

  def getY(self,xu):
    for i in range(0, len(self.x)-1):
      if self.x[i] <= xu <= self.x[i+1]:
        h = xu - self.x[i]
        return self.c[0][i]*(h**3) + self.c[1][i]*(h**2) + self.c[2][i]*h + self.c[3][i]

  def savePlot(self,filename,m):
    xu = self._frange(self.x[0],self.x[-1],m)
    yu = [self.getY(xxu) for xxu in xu]
    plt.plot(xu, yu)
    plt.plot(self.x, self.y, 'co')
    plt.tight_layout()
    plt.savefig(filename + '.png')
    plt.close('all')
    with open(filename + '.txt', 'w') as f:
      for (xxu, yyu) in zip(xu, yu):
          f.write("%.20f %.20f\n" % (xxu, yyu))

  def _getTridiagSys(self,h,dif):
    e = h[1:-1]
    f = [(a+b)*2.0 for a,b in zip(h[:-1], h[1:])]
    g = h[1:-1]
    r = [(a-b)*6.0 for a,b in zip(dif[1:], dif[:-1])]
    return (e,f,g,r)

  def _getResTridiagSys(self,e,f,g,r):
    for i in range(0,len(e)):
      e[i] = e[i] / f[i]
      f[i+1] = f[i+1] - e[i] * g[i]
    for i in range(0,len(e)):
      r[i+1] = r[i+1] - e[i] * r[i]
    n = len(f)
    s2d = [0.0] * n
    s2d[n-1] = r[n-1]/f[n-1]
    for i in range(n-2,-1,-1):
      s2d[i] = (r[i] - g[i] * s2d[i+1])/f[i]
    return s2d

  def _getCoeficients(self):
    h = [a-b for a,b in zip(self.x[1:],self.x[:-1])]
    dif = [(a-b)/c for a,b,c in zip(self.y[1:],self.y[:-1],h)]
```

```
        (e,f,g,r) = self._getTridiagSys(h,dif)
        s2d = [0.0] + self._getResTridiagSys(e,f,g,r) + [0.0]
        c4 = self.y[:-1]
        c2 = [a/2.0 for a in s2d[:-1]]
        c1 = [(a-b)/(c*6.0) for a,b,c in zip(s2d[1:],s2d[:-1],h)]
        c3 = [a-((b+c*2.0)/6.0)*d for a,b,c,d in zip(dif,s2d[1:],
            s2d[:-1],h)]
        return (c1,c2,c3,c4)

    def _frange(self,head,tail,m):
        l = [head] * m
        jump = (tail-head)/(m-1)
        for i in range(1, m-1):
            l[i] = l[i-1] + jump
        l[m-1] = tail
        return l
```