CS 323 Numerical Analysis and Computing

Programming Assignments 2 & 3

Due: 04/02/2018

Important

- Sakai submission only.
- No late submission will be accepted.
- Only **Python** or **Matlab** implementation will be accepted. No credit will be given for implementation in other languages.
- This is an individual assignment.
- **Submit your source code** - no screenshot, or copy/paste into a word/pdf document.
- Follow the instructions given in each of the problems.
- In each question, if your code does not run, no credit will be given for the respective problem and the solutions you got from it.
- Write a short report (this is problem 4), presenting only the requested information. Do not just paste all the program output.
- You have to submit 4 files (3 source codes, and 1 pdf file)

(1.) (20 points) Write a function named **NaturalSpline.m**, if using **Matlab**, or **NaturalSpline.py**, if using **Python**, to implement the following algorithm to construct the **Natural Cubic Spline** interpolant $S(x)$, defined at the numbers $x_0 < x_1 < \cdots < x_n$.

Remember that

$$S(x) = S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \quad \text{for } x_j \le x \le x_{j+1}$$

Algorithm – Natural Cubic Spline

INPUT $n; x_0, x_1, \ldots, x_n; a_0 = f(x_0), a_1 = f(x_1), \ldots, a_n = f(x_n)$

OUTPUT a_j, b_j, c_j, d_j for $j = 0, 1, \ldots, n-1$

Step 1 For $i = 0, 1, \ldots, n-1$ set $h_i = x_{i+1} - x_i$

Step 2 For $i = 1, 2, \ldots, n-1$ set

$$\alpha_i = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1})$$

Step 3 Set $l_0 = 1; \quad \mu_0 = 0; \quad z_0 = 0$

Step 4 For $i = 1, 2, \ldots, n-1$ set

$$\begin{aligned} l_i &= 2(x_{i+1} - x_{i-1}) - h_{i-1}\mu_{i-1}; \\ \mu_i &= h_i/l_i; \\ z_i &= (\alpha_i - h_{i-1}z_{i-1})/l_i; \end{aligned}$$

Step 5 Set $l_n = 1; \quad z_n = 0; \quad c_n = 0$

Step 6 For $j = n-1, n-2, \ldots, 0$ set

$$\begin{aligned} c_j &= z_j - \mu_j c_{j+1}; \\ b_j &= (a_{j+1} - a_j)/h_j - h_j(c_{j+1} + 2c_j)/3; \\ d_j &= (c_{j+1} - c_j)/(3h_j); \end{aligned}$$

Step 7 OUTPUT $(a_j, b_j, c_j, d_j$ for $j = 0, 1, \ldots, n-1)$
STOP

(2.) (10 points) Write a function named **DividedDifferences.m**, if using **Matlab**, or **DividedDifferences.py**, if using **Python**, to obtain the divided-difference coefficients of the interpolation polynomial P_n on the $(n+1)$ distinct points $x_0, x_1, \ldots, x_n$ for the function f.

Algorithm – Newton's Divided-Differences

INPUT $x_0, x_1, \ldots, x_n$; and $f(x_0), f(x_1), \ldots, f(x_n)$ as $F_{0,0}, F_{1,0}, \ldots, F_{n,0}$

OUTPUT $F_{0,0}, F_{1,1}, \ldots, F_{n,n}$, where

$$P_n(x) = F_{0,0} + \sum_{i=1}^{n} F_{i,i} \prod_{j=0}^{i-1} (x - x_j)$$

and $F_{i,i} = f[x_0, x_1, \cdots, x_i]$

Step 1 For $i = 1, 2, \ldots, n$
  For $j = 1, 2, \ldots, i$
    set $F_{i,j} = \dfrac{F_{i,j-1} - F_{i-1,j-1}}{x_i - x_{i-j}}$ (Notice $F_{i,j} = f[x_{i-j}, \cdots, x_i]$.)

Step 2 OUTPUT $(F_{0,0}, F_{1,1}, \ldots, F_{n,n})$
  STOP

(3.) (15 points) Figure 1 shows a ruddy duck in flight. To approximate the top profile of the duck, we have chosen points along the curve through which we want the approximating curve to pass.

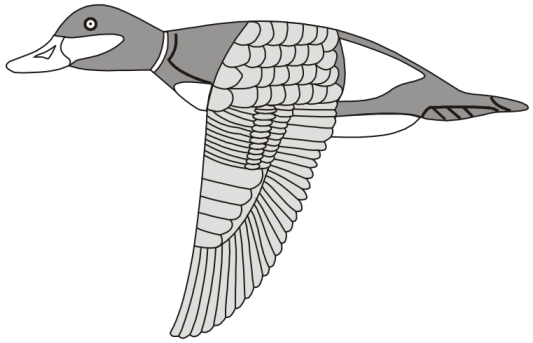

Figure 1: Flying duck.

The tables below list the coordinates of 21 data points relative to the superimposed coordinate system shown in Figure 2. Notice that more points are used when the curve is changing rapidly than when it is changing more slowly.

x	0.9	1.3	1.9	2.1	2.6	3.0	3.9	4.4	4.7	5.0	6.0
f(x)	1.3	1.5	1.85	2.1	2.6	2.7	2.4	2.15	2.05	2.1	2.25

x	7.0	8.0	9.2	10.5	11.3	11.6	12.0	12.6	13.0	13.3
f(x)	2.3	2.25	1.95	1.4	0.9	0.7	0.6	0.5	0.4	0.25

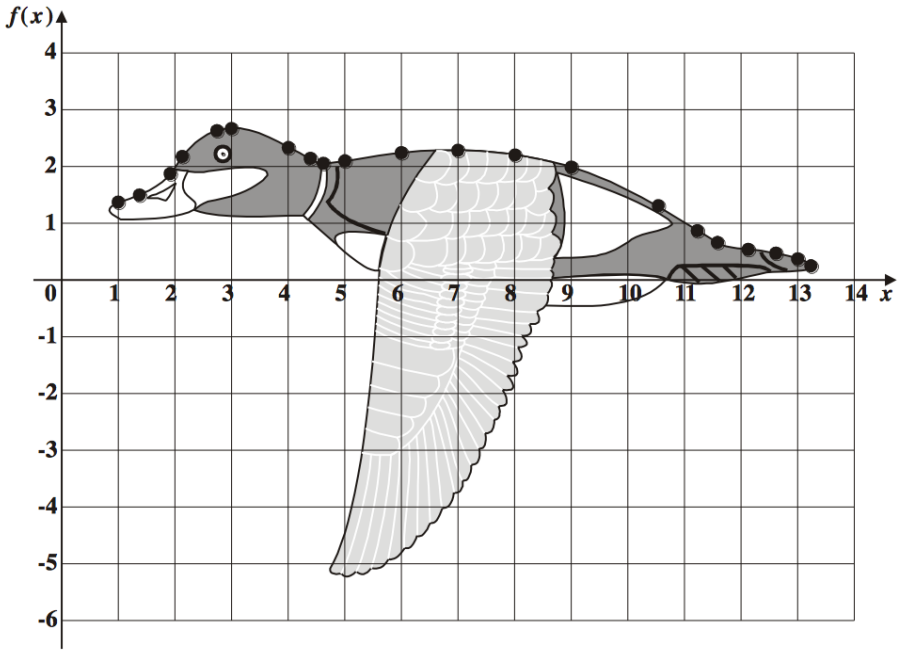

Figure 2: Points through which we want the approximating curve to pass.

Instructions for Problem 3:

Write a script, named **Problem3.m**, if you are using Matlab, or **Problem3.py**, if you are using Python, that

(a) uses your code from Problem 1 to generate the coefficients for the natural cubic spline going through the given data points;

(b) uses your code from Problem 2 to generate the coefficients for the interpolating polynomial (in Newton form) going through the given data points;

(c) plots, in the same figure, in a coordinated system:

* the given data points
* the cubic spline
* the interpolating polynomial

Note: For the figure, use different colors for each plot, and plot the data points with a big dot (or circle, or asterisk) so it is easy to identify the given points.

(4.) (15 points) Write a "report". Your report should be named Report.pdf and should include:

(a) a table with the coefficients obtained for the natural cubic spline in Problem 3

(b) a table with the coefficients obtained for the interpolating polynomial in Problem 3

(c) the figure from Problem 3