Graded Homework 3 Foundations of Computational Math 2 Spring 2025

The solutions are due by 11:59PM on Friday March 21, 2025

The due date is 1 week after Spring Break and therefore includes 2 weeks of course in session. You are encouraged, but not required, to get the written questions and a portion of the coding done before Spring Break. For example getting the piecewise polynomial code and the first spline code debugged and compared to each other and the global polynomial based on Chebyshev points that you have from the previous programming assignment completed before Spring Break is recommended.

Written Exercises

Problem 3.1

Assuming that the nodes are uniformly spaced, we have derived the form of the cubic B-spline $B_{3,i}(t)$ and determined its values and the values of $B'_{3,i}(t)$ and $B''_{3,i}(t)$ at the nodes t_{i-2}, t_{i-1}, t_{i-1}, t_i, t_{i+1}, and t_{i+2}. We also derived $B_{1,i}(t)$ and saw that it was the familiar hat function.

3.1.a. Derive the formula of the quadratic B-spline $B_{2,i}(t)$ and determine its values and the values of $B'_{2,i}(t)$ and $B''_{2,i}(t)$ at the appropriate nodes.

3.1.b. Derive the formula of the quintic B-spline $B_{5,i}(t)$ and determine its values and the values of $B'_{5,i}(t)$ and $B''_{5,i}(t)$ at the appropriate nodes.

Problem 3.2

Consider an interpolatory quadratic spline, $s(x)$, that satisfies the following interpolation conditions and single boundary condition:

$$s(x_i) = f(x_i) = f_i, \;\; 0 \le i \le n$$

$$s'(x_0) = f'(x_0) = f'_0$$

where the x_i are distinct.

3.2.a. Derive a linear system of equations that yields the values

$$s'(x_i) = s'_i \;\; 0 \le i \le n$$

that are used as parameters to define the quadratic spline $s(x)$.

3.2.b. Identify important structure in the linear system and show that it defines a unique quadratic spline.

3.2.c. Use the structure of the system to show that if $f(x)$ is a quadratic polynomial then $s(x) = f(x)$.

Problem 3.3

For this problem, consider $f(x) = 2 - x^2$ on $[-1,\ 1]$.

3.3.a. Determine the constant minimax approximation, $p_0(x)$.

3.3.b. Determine the linear minimax approximation, $p_1(x)$.

3.3.c. Derive the linear near-minimax polynomial approximation, $c_1(x)$.

3.3.d. Explain the results.

Programming Exercise

Codes

1. **Interpolating polynomial**: The routine is for a single interpolating polynomial, $p_n(x)$, on $[a,b]$ that uses the Barycentric form 1 and Chebyshev points of the first or second kind and therefore converges. Note you must map the Chebyshev points fromthere values in $[-1.1]$ to $[a,b]$ to get the mesh points x_j with which the interpolating polynomial is generated. You should have this code from the previous programming assignment.

2. **Piecewise interpolating polynomial**: The routine for the piecewise interpolating polynomial $g_d(x)$, on $[a,b]$ where the degree s should support chosing d to be 1 or 2 or 3 and and interpolating the function values on each subinterval $[a_i,b_i]$ at the end points and the appropriate number and placement of additional interpolating points in (a_i,b_i). You should also support the option of a piecewise cubic Hermite interpolating polynomial where on each subinterval $[a_i,b_i]$ the local cubic Hermite polynomial interpolates $f(a_i)$, $f(b_i)$, $f'(a_i)$, $f\prime(b_i)$. Your code should support a uniform mesh and the use of Chebyshev points of the second kind within each subinterval $[a_i,b_i]$. It is suggested that using a local Newton divided difference form is the way to support these requirements since it easily incorporates the Hermite cubic with the other standard interpolants on $[a_i,b_i]$.

3. **Interpolatory cubic spline**: In addition to the mesh points and the function $f(x)$ this code must accept as input the specification of two boundary conditions to be applied in the formation of $s(x)$. You must implement two spline codes. Spline Code 1 may use either the s_i' parameterization or the s_i'' parameterization. Spline Code 2 must the parameterization in terms of the cubic B-spline basis.

4. Implement a code that evaluates the spline $s(x)$ and piecewise polynomial $g_d(x)$ produced by your codes. You should already have a code that evaluates the interpolating polynoial $p_n(x)$.

5. Implement the supporting code required to empirically validate the correct functioning of your codes and the tasks below.

Code Comments:

1. Your codes should be able to run double precision (assumed to be IEEE standard FP).

2. Your codes must be efficient in time and space and make sure you discuss these aspects of your implementations. Include a discussion of how you represent and evaluate the local polynomials defining $g_d(x)$ in your piecewise polynomial code.

3. You may use the parameterization of $s(x)$ in terms of the s_i'' or s_i' for Spline Code 1. This code must must accept data on a **nonuniform mesh**.

4. Spline Code 2 based on the B-spline basis may be restricted to a uniform mesh if you wish but you may implement the nonuniform mesh at your discretion.

5. Both spline codes should support both of the basic Hermite boundary conditions. That is, either the pair of values $(s''(x_0), s''(x_n))$ or the pair of values $(s'(x_0), s'(x_n))$ are specified. Depending on your choice of parameterization one of these cases may require the addition of two extra equations needed to specify boundary conditions in terms of the appropriate s''_i or s'_i parameters. Your solution should include the derivation of these equations. For the Spline Code 2 based on the B-spline basis you should also explain how you handle the two types of boundary conditions.

Tasks

Task 1

Empirically validate the correct functioning of your code.

- You must design experiments and describe the outcomes that provide evidence that your code is working. Evidence must also be given that you code works correctly for each of the required boundary conditions.

- This should include running your code to approximate carefully selected functions $g(x)$ for which the results are known. For example, if $g(x)$ is any polynomial of degree $d \leq 3$, the spline, $s(x)$, should reproduce its value at any x. Similarly, if $g(x)$ is a piecewise cubic with intervals defined by the same mesh as $s(x)$ then $s(x)$ should match $g(x)$ for any x between x_0 and x_n.

- Compare the functioning of the two spline codes to each other and, when appropriate for the test function, to the piecewise polynomial code. Be careful when comparing the different functions. For example make sure the subintervals $[a_i, b_i]$ are the same for any piecewise polynomial and spline. The rest of the interpolating points can be defined inside those subintervals depending on the degree of the piecewise polynomial. Compare accuracy for the global interpolating polynomial, the splines and piecewise polynomial and compare the different levels of complexity (operations and space) required to reach a particular level of error.

- Once this has been done for specific functions, randomly generating similar functions and summarizing the results can produce good evidence of correct functioning.

- You can, but are not required to so do, also check you spline results against other libraries. **Make sure however that you know exactly what the other library is choosing for boundary conditions etc. You must compare splines that are known to be the same in exact arithmetic.** Care should also be taken with respect to what you consider the "answer" for the other library's results. If you print them and then copy to your code you have modified the results so that they print in ASCII format. This is not the answer's full single or double precision representation. The most reliable way is to output to a file raw binary IEEE single or double precision results and read them into your code using the appropriate format in the high-level language of your code. This is a technique of input/output that you should develop skill in using. You are not required to use it here but make sure you compare an appropriately precise output value if you use file output to compare your results to a library code.

Task 2

Suppose you have functions $y(t)$, $f(t)$, and $D(t)$ related as follows:

$$D(t) = e^{-ty(t)} = e^{-\int_0^t f(\tau)d\tau}$$

$$\therefore \quad f(t) = y(t) + ty'(t)$$

As a result, given any of one of the functions, $D(t)$, $f(t)$ or $y(t)$ we can recover the other two. Suppose, in practice $y(t)$ is available as data in the form of discrete values of (t_i, y_i), for $0 \leq i \leq n$ rather than as a continuous function. Specifically, consider the following data (t_i, y_i):

t_i	0.5	1.0	2.0	4.0	5.0	10.0	15.0	20.0
$y(t_i)$	0.04	0.05	0.0682	0.0801	0.0940	0.0981	0.0912	0.0857

Note the mesh in t is nonuniform.

Use a natural boundary condition interpolatory cubic spline, $s(t)$, based on the data (t_i, y_i) to estimate $y(t)$, $f(t)$ and $D(t)$. Tabulate your estimates from $t_1 = 0.5$ to t_{40} with increment $\Delta t = 0.5$.

The piecewise polynomial $g_d(x)$ from your code is not continuously differentiable in general. Investigate the use of $g_d(x)$ rather than a cubic spline for this Task. If it is possible, what must you do differently to generate estimates for $y(t)$, $f(t)$ and $D(t)$ and how does the choice of degree affect your results compared to a cubic spline?