Program 3 Foundations of Computational Math 1 Fall 2020

Due date: 11:59PM on November 6, 2020

Programming Exercise

Codes

1. **Piecewise interpolating polynomial**: The routine for the piecewise interpolating polynomial $g_s(x)$, on $[a, b]$ where the degree s should support chosing s to be 1 or 2.

2. **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 will use either the s_i' parameterization or the s_i'' parameterization. Spline Code 2 must use the parameterization in terms of the cubic B-spline basis.

3. Implement a code that evaluates the spline $s(x)$ and piecewise polynomial $g_s(x)$ produced by your codes.

4. 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 in single or 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 $g_s(x)$ 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.
- Once this has been done for specific functions, randomly generating similar functions and summarizing the results can produce good evidence of correct functioning.
- The rate of convergence can be estimated empirically from estimated values of $\|f(x) - s(x)\|_\infty$ as $h \to 0$. So if you have an error estimate $E(h)$ and $E(h/2)$ based on a finer grid with the intervals halved, then the rate of convergence can be estimated by

$$\log_2 \frac{|E(h)|}{|E(h/2)|}$$

which should converge to 4 for the spline codes. You can estimate the rate of convergence for the first and second derivative as well. See Table 8.5 on p.

361 of the textbook for an example estimation of cubic spline convergence rate. You should be able to reproduce it and similar results along with assessing the convergence of the piecewise polynomial code.

- You can 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 $G(t)$ related as follows:

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

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

As a result, given any of one of the functions, $G(t)$, $f(t)$ 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 $G(t)$. Tabulate your estimates from $t_1 = 0.5$ to t_{40} with increment $\Delta t = 0.5$.

The piecewise polynomial $g_s(x)$ from your code is not continuously differentiable in general. Investigate the use of $g_s(x)$ 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 $G(t)$ and how does the choice of degree affect your results compared to a cubic spline?

Other Test Problems

After you have submitted you solutions you should make an appointment with the TA Joshua Gonzalez. He will ask you to demonstrate your code on some test problems as a final evaluation of your codes correctness. So your codes should be able to read the mesh points and function values (x_i, f_i), for $i = 0, \ldots, n$, the type of boundary condition and the associated values from a formatted problem input file. (The format for the input file will be posted on the class website.) You should also be prepared to explain the design and operation of your code to him.

Solving the Linear Systems

Construction of the splines requires the solution of a linear system of equations that is tridiagonal or nearly tridiagonal. You may apply any numerical linear algebra library you choose, e.g., Matlab system solvers, LAPACK, to find a subroutine to solve such systems or implement your own. For tridiagonal matrices Thomas' algorithm is described in the textbook. This could also be used as a piece of an algorithm to solve systems with matrices that are tridiagonal in all but a small number, $O(1)$, number of rows. You must document your approach and cite references appropriately.

Submission of Results

Expected results comprise:

- A document describing your solutions as prescribed in the notes on writing up a programming solution posted on the class website.
- The source code, makefiles, and instructions on how to compile and execute your code including the Math Department machine used, if applicable.
- Code documentation should be included in each routine.
- All text files that do not contain code or makefiles must be PDF files. **Do not send Microsoft word files of any type**.

These results should be submitted by 11:59 PM on the due date.