Homework 3 Introduction to Computational Finance Spring 2023

Solutions due Friday March 10, 2023
Note that this due date is the day after the exam. Working on this assignment is an effective part of studying for the exam. Do not wait until the last minute to work on this program.

Answers to the homework problems and programming tasks should be submitted using the class canvas page.

You should submit pdf files. **Do not sent Word files or any other text processing tool's input file**.

As with all homework assignments you are allowed and encouraged to consult the relevant literature. You are also expected **to cite all literature that is used to generate your solutions and your solutions must make clear your understanding of the work cited.**

Programming Assignment

Algorithms Required:

Assume you are given distinct points $a = x_0, \ldots, x_n = b$, on the interval $[a, b]$ and a function $f(x)$ defined on the interval. Implement the following algorithms

1. **Newton form of the interpolating polynomial**:. There should be two routines. The first computes the required divided differences and returns them for use by the second which evaluates the Newton form of the polynomial for a given x value. You may find it useful to implement this so that a set of x values can be given as input and the set of the values of the polynomial at those points is returned. In your solutions denote this polynomial $N_d(x)$ when it has degree d. This first routine should accept any mesh of x_i points on the interval of interest, i.e., it is not restricted to any specific set of x_i values like Chebyshev points or equidistant points. The second routine should accept any set of x values on which the polynomial is to be evaluated including points on the mesh used to generate the divided differences.

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

Code Comments:

- Your codes should be able to run in single or double precision (assumed to be IEEE standard FP).
- Your codes must be efficient in time and space and make sure you discuss these aspects of your implementations.

- Compute the divided differences using the divided difference table approach. Your implementation **must** be efficient in storage, i.e., $O(n)$ space. Therefore, you must decide what form of $N_d(x)$ you will use, i.e., the one starting with f_0 and adding the values at $x_1, \ldots, x_{n-1},\ x_n$ or the one starting with f_n and adding the f_i values at $x_{n-1}, \ldots, x_1,\ x_0$. In either case when computing the divided difference table your algorithm must use only $O(n)$ work space and should not store the entire table.

- The piecewise interpolating polynomial algorithm should use a simple selection mechanism for the degree, i.e., do not write separate routines for each degree in some assumed range of degrees.

- The piecewise interpolating polynomials must be continuous. So the mesh of interpolation points used in each subinterval must include the two end points. Therefore, the piecewise linear polynomial, $g_1(x)$, will define a linear polynomial, $p_{1,i}(x)$ for each of the n intervals, $[x_i, x_{i+1}]$, $i = 0, \ldots, n-1$. The piecewise quadratic polynomial, $g_2(x)$, will define a quadratic polynomial, $p_{2,i}(x)$ for each of the $n/2$ intervals with $n = 2k$, i.e., n is assumed to be even. The intervals are $[x_{2i}, x_{2i+2}]$, $i = 0, \ldots, (n/2) - 1$. Note that the points with the odd indices are contained in these intervals.

- The low degree of $s = 1,\ 2$ implies that 1 or 2 divided differeces are required per interval. You may implement this simple computation separately from the Newton divided difference polynomial code above that must handle a much wider range of degree and table size.

- The piecewise polynomial codes must be able to specify a general mesh of points and a uniform mesh of points.

When investigating the performance and correctness of your codes:

- You should carefully choose the functions to interpolate and demonstrate the predicted error holds for both global and piecewise interpolants.

- Recall the error should be very near zero at the interpolation points.

- To approximate $||f(x) - g_s(x)||_\infty$ and $||f(x) - N_d(x)||_\infty$, sample $f(x)$ and the approximating function at a large number of points, z_i, in the interval and take the largest value of the magnitude of the difference between $f(z_i)$ and the approximating function at z_i. When the number of z_i is sufficiently large this will give a good approximation of the error.

Tasks:

Task 1 Discuss the complexity of preprocessing and evaluation for each method. Note this complexity analysis should include the search for the interval in which the value resides for piecewise interpolating and how assumptions about the mesh can influence the complexity of the search.

Task 2 It is known from the error bounds that $N_d(x)$ is identical to $f(x)$ as long as $f(x)$ is a polynomial of degree less than or equal to d. Use this as one debugging/validation test for your Newton interpolating polynomial code by producing $N_d(x)$ for $d = 1, 2, \ldots c$ where $f(x)$ is a polynomial of degree c. Do this for many such $f(x)$ and present the results in such a way that it is convincing that your code reproduces polynomials correctly.

Task 3 Apply the error bounds of the notes for piecewise polynomial interpolation with degrees $1 \leq s \leq 2$ to determine the subinterval sizes that guarantee a desired error $\|f - g_s\|_\infty$ and verify this by experiment. Comment on whether the experiments indicate that the error bounds are tight or loose for the particular functions you choose to interpolate.

Task 4 Empirically investigate the convergence and divergence to $f(x)$ for: interpolation with a single polynomial and interpolation with the piecewise polynomials as the number of points increases in terms of the infinity norm of each of the interpolation approaches.

It is known that divergence will occur for a uniform mesh and some functions. Make sure that you present your results in a convincing manner for these cases, e.g., the one in the notes. Avoid brakdown of your code, e.g., generating overflow, as the polynomials diverge from $f(x)$. You can compute $\|f - g_s\|_\infty$ and $\|f - N_d(x)\|_\infty$ for each mesh and stop when it is clear that there is a trend of increasing error.

Functions:

For the tasks, your experiments should include, as appropriate, the following functions on $[-1, 1]$ from the notes

$$f(x) = |\alpha x| + \frac{x}{2} - x^2$$

$$f(x) = \frac{1}{1 + \alpha x^2}$$

You **must** also consider other functions to demonstrate or evaluate hypotheses and properties of the routines in your solution.

Meshes:

You should include uniform spacing of the interpolating points that allows the generation of a uniform spacing of the x_i for any n, i.e., $x_i - x_{i-1} = (b - a)/n$ defines the family of uniform meshes as a function of the number of points $n + 1$.

You should include the $n+1$ roots of the Chebyshev polynomial of degree $n+1$, $T_{n+1}(x)$, which are known and in the interval $-1 \leq x \leq 1$ to be

$$x_k = \cos\left(\frac{(2k+1)}{(n+1)}\,\frac{\pi}{2}\right), \quad 0 \leq k \leq n$$

These are called the Chebyshev points of the first kind.

You should include the $n+1$ points called the Chebyshev points of the second kind in the interval $-1 \le x \le 1$ given by

$$x_k = \cos\left(\frac{(k\pi)}{n}\right), \quad 0 \le k \le n.$$

Demonstration of Codes:

After submitting your solutions, you may be asked to apply you code to meshes and functions chosen by the instructor to demonstrate correctness.

Written Exercises

Problem 3.2

Suppose we want to approximate a function$f(x)$ on the interval $[a, b]$ with a piecewise quadratic interpolating polynomial, $g_2(x)$, with a constant spacing, h, of the interpolation points $a = x_0 < x_1 \ldots < x_n = b$. That is, for any $a \le x \le b$, the value of $f(x)$ is approximated by evaluating the quadratic polynomial that interpolates f at x_{i-1}, x_i, and x_{i+1} for some i with $x = x_i + sh$, $x_{i-1} = x_i - h$, $x_{i+1} = x_i + h$ and $-1 \le s \le 1$. (How i is chosen given a particular value of x is not important for this problem. All that is needed is the condition $x_{i-1} \le x \le x_{i+1}$.)

Suppose we want to guarantee that the **relative error** of the approximation is less than 10^{-d}, i.e., d digits of accuracy. Specifically,

$$\frac{|f(x) - g_2(x)|}{|f(x)|} \le 10^{-d}.$$

(It is assumed that $|f(x)|$ is sufficiently far from 0 on the interval $[a, b]$ for relative accuracy to be a useful value.) Derive a bound on h that guarantees the desired accuracy and apply it to interpolating $f(x) = e^x \sin x$ on the interval $\frac{\pi}{4} \le x \le \frac{3\pi}{4}$ with relative accuracy of 10^{-4}. (The sin is bounded away from 0 on this interval.)

Compare your predicted accuracy to the accuracy you achieve by forming $g_2(x)$ for h's that satisfy your bound and h's that do not.

Problem 3.3

Let $p_3(x)$ be the unique polynomial that interpolates the data

$$(x_0, f_0), \quad (x_1, f_1), \quad (x_2, f_2), \quad (x_3, f_3)$$

and let $\tilde{p}_3(x)$ be the unique polynomial that interpolates the data

$$(x_1, f_1), \quad (x_2, f_2), \quad (x_3, f_3), \quad (x_4, f_4)$$

Suppose that you have stored a representation of $p_3(x)$ based on divided differences, i.e., you have stored a vector **based at x_0 for a degree 3 polynomial**

$$INFO(x_0, 3) = \begin{bmatrix} f_0 \\ f[x_0, x_1] \\ f[x_0, x_1, x_2] \\ f[x_0, x_1, x_2, x_3] \end{bmatrix}$$

and you have the values of $x_0, \quad x_1, \quad x_2, \quad x_3$ stored.

Note you only have the information in INFO and the 4 values of x_i. You do not have any other f_i values or any other divided differences available unless explicitly stated in the questions below.

3.3.a. Suppose that in addition to the vector INFO you are given the the new pair $(x_4, \ f_4)$. Describe an algorithm that updates the vector INFO to contain the information needed to specify $\tilde{p}_3(x)$ in the same manner the initial contents of INFO, i.e., $INFO(x_0, 3)$, specified $p_3(x)$. That is you want to compute $INFO(x_1, 3)$ which specifies the cubic $\tilde{p}_3(x)$.

You should make your algorithm as efficient as possible in both storage and number of operations.

3.3.b. What is the compexity of your algorithm, **in both storage and number of operations**, when applied to a pair of polynomials of degree n, i.e., is it $O(n^2)$ or $O(n)$? Justify your answer.