Graded Homework 1 Foundations of Computational Math 2 Spring 2022

The solutions are due by 11:59PM on Monday 7 February, 2022

Programming Exercise

Algorithms Required

You will need the following algorithms for this assignment. Note that the class notes and papers posted on the class webpage have pseudocode for many of these functions.

- Interpolation by Barycentric Form 2 of Lagrange interpolating polynomial that can use any of three meshes: uniform and Chebyshev points of the first kind and Chebyshev points of the second kind. This requires two routines and they should be able to work in IEEE double or single precision as required by the experiments.
  - A routine to evaluate the coefficients β_i, $0 \leq i \leq n$, required by the Lagrange Barycentric Form 2 should be implemented using $O(n)$ space the appropriate reduced number of operations depending on the mesh used as described in the class notes. The routine should also evaluate the function values $f(x_i)$, $0 \leq i \leq n$. The input to this routine includes a flag indicating which of the three mesh choices is to be used, n, and the function $f(x)$. The output should be the β_i and $f_i = f(x_i)$ values.
  - A routine to evaluate $p_n(x)$ in Barycentric Form 2. It may be useful to give a vector of x values at which you need $p_n(x)$ rather than calling the routine once for each value. Be sure to address the issues when $fl(x - x_i)$ is 0 or very small as discussed in the papers by Higham, and Berrut and Trefethen on the class webpage. (Note there is a typo in proposed code solution in the latter paper.)
- A routine to evaluate the divided differences required for the Newton form of an interpolating polynomial $p_n(x)$ using $O(n^2)$ operations and $O(n)$ space. The routine shouldcompute the function values $f(x_i)$, $0 \leq i \leq n$ and the divided differences that are required. You may use any of the algorithms we have discussed to compute the divided differences. The output should be the divided differences and $f_i = f(x_i)$ values.
- A routine based on the adapted Horner's rule to evaluate a polynomial $p_n(x)$ defined in terms of the Newton basis. (Note this also makes it possible to evaluate the monomial basis by taking all x_i to be the same value or any set of x_i with all or some of the values repeated.) It may be useful to give a vector of x values at which you need $p_n(x)$ rather than calling the routine once for each value.

- A routine to order a set of distinct mesh points x_i, $0 \leq i \leq n$ into increasing order $x_0 < x_1 < \cdots < x_{n-1} < x_n$, or decreasing order $x_0 > x_1 > \cdots > x_{n-1} > x_n$, or satisfying the Leja ordering (see Set 8 slide 34 of the class notes and Study Questions Set 1). The input includes the unordered x_i and a flag indicating the desired ordering. The output should be the ordered x_i.

- A routine to evaluate $\|r(x)\|_\infty$ and related statistics, e.g., maximum, average etc.. To approximate $||r(x)||_\infty$, the function should be evaluated at a large number of points in the interval of interest and the largest magnitude returned. This will be applied to various functions, e.g., $r(x) = p_n(x) - \hat{p}_n(x)$ where $p_n(x)$ is the "exact" value of the interpolating polynomial and $\hat{p}_n(x)$ is the computed value or the "exact" value of a perturbed interpolation polynomial. The computation of the mean and variance of the values of $r(x)$ or other statistics may be useful in your empirical analysis and presentation.

Comments on Routines and Experiments

- The tasks require the systematic empirical evaluation of many cases of parameter choices and summarizing them to make conclusions on code correctness, accuracy, stability, and conditioning. It is absolutely crucial that you organize your computations using scripts and parameterized codes etc. to automate the process so as not to be overwhelmed with "manual" editing, compiling and manipulating data. This is an important skill to master for computational mathematics of any type.

- 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.

- All experiments assessing accuracy and stability will be for the single precision execution of the codes. Double precision execution will be used when generating "exact" values needed for computing the error when analyzing the accuracy and stability of the single precision codes.

- When applying Chebyshev points of the first kind as the mesh, it is assumed the interval of interest is $[-1, 1]$. This is clearly not the only interval on which you will assess, for example, accuracy. So be sure to develop the code to use the appropriate change of variables.

- You will be assessing empirically the stability, and accuracy of interpolating polynomials of various degrees on various intervals for various functions. This will require generating many values such as errors, differences in exact and computed values, stability and error bounds, and condition numbers are many values of the independent variable. Carefully consider how you are going to present the data using selected examples, curves, histograms, statistics and bounds. Pages of tables and brief descriptions of what you see in the data are not acceptable.

- For some of the functions that are polynomials given below, a product form is available. This form can be evaluated using the simple incremental product evaluation

$$p_n(x) = \alpha_n(x - \rho_1) \cdots (x - \rho_n) \tag{1}$$

given by:

$d_0 = \alpha_n$
for $i = 1 : n$
    $d_i = d_{i-1} * (x - \rho_i)$
end

$p_n(x) = d_n$

This algorithm can be shown to compute $p_n(x)$ to high relative accuracy (Higham 2002 Accuracy and Stability of Numerical Algorithms, Second Edition). Specifically,

$$d_n = p_n(x)(1 + \mu), \qquad |\mu| \leq \gamma_{2n+1}$$

where $\gamma_k = ku/(1 - ku)$ and u is the unit roundoff of the floating point system used. his "exact" form can be used when evaluating accuracy or stability of an algorithm. If such an alternate form of $p_n(x)$ or $f(x)$ is not available then using double precision will be acceptable as "exact" when assessing the error and stability of a single precision code for a well-conditioned or moderately ill-conditioned problem is considered.

- For some of the $f(x)$ that are polynomials given below the monomial form is also available. These should be useful in assessing the Horner's rule routine.

Functions of Interest

You can use any functions that you think appropriate for evaluating the accuracy and reliability of your codes and that approaches they implement, however, some suggestions are given below. The first 3 functions are polynomials, $p_d(x)$, that can be used to define interpolation problems using the data points (x_i, y_i), $0 \leq i \leq d$, where d is the degree of the polynomial and $y_i = p_d(x_i)$ for the mesh points of interest. Note that $f_3(x) = \ell_n(x)$ satisfies the same canonical interpolation problem for any mesh and was used in Higham's IMA Journal of Numerical Analysis 2004 paper as a problem that distinguished the behavior of Barycentric Forms 1 and 2 on a uniform mesh with 30 points. Note that the polynomials can also be used with meshes that have many more points than $d + 1$ in order to validate the correctness of your codes by considering the error observed.

For $f_2(x)$ and $f_3(x)$ you may choose different degrees when performing the tasks below. The function $f_4(x)$ was also used with a uniform mesh of 30 points in Higham's paper.

- Function 1

$$f_1(x) = (x-2)^9$$
$$= x^9 - 18x^8 + 144x^7 - 672x^6 + 2016x^5 - 4032x^4 + 5376x^3 - 4608x^2 + 2304x - 512$$

- Function 2 (parameterized by degree d)

$$f_2(x) = \prod_{i=1}^{d} (x-i)$$

- Function 3 (parameterized by degree n)

$$f_3(x) = \ell_n(x)$$
$$f_3(x_i) = 0, \quad 0 \leq i \leq n-1$$
$$f_3(x_n) = 1$$

  where $\ell_n(x)$ is a Lagrange basis function.

- Function 4

$$f_4(x) = \frac{1}{1+25x^2}$$

Tasks

The tasks below should be carried out using the routines described above on the functions listed above and any other functions you think useful. Choose intervals of interpolation $[a, b]$ that contain all of the defining points of the functions, e.g., the roots of the polynomial. The intervals on which you assess stability and accuracy can be subintervals of $[a.b]$ and, e.g., **need not** contain all of the roots of the polynomial or all of the mesh points. These subintervals should, of course, be as large as computationally tractable but should also be of interest, e.g., intervals where there is variation in the behavior of f or the interpolating polynomial.

Task 1

Describe the design of your codes and discuss the complexity with respect to time and space. Empirically validate your routines. Your arguments for the correctness of your codes may include referencing their behaviors on the later tasks if appropriate, but your write up for this task should summarize those behaviors leaving the details for the write up of the later tasks.

Task 2, 3 and 4

Task 2 performs the subtasks described below for f_1, Task 3 performs the subtasks described below for f_2, and Task 4 performs the subtasks described below for f_3.

For the function associated with the task on intervals perform the following subtasks:

1. Consider the interpolating problem that the given polynomial solves on the uniform mesh points and Chebyshev points of the first kind, i.e., $y_i = f(x_i)$, $0 \leq i \leq m$ where m is 9 for $f_1(x)$, d for $f_2(x)$ and n for $f_3(x)$. For $f_2(x)$ and $f_3(x)$ choose at least two different degrees that are greater than 20 for each mesh type. For $f_3(x)$ include $n = 29$ (30 points) to compare to the results in Higham's paper.

2. Assess the accuracy and stability of the single precision codes using the appropriate bounds from the notes and literature on the class webpage. (As described earlier, "exact" values of the interpolating polynomial for accuracy and stability assessment should be done in double precision.)

   This should be done for

   - Barycentric form 2 of the polynomial
   - Newton form with the mesh points in increasing order, decreasing order and satisfying the Leja ordering conditions.

   You should provide plots similar to those in Higham's paper for a small number of illustrative examples with 30 points for the uniform mesh and the Chebyshev points of the first kind. The other results should be summarized to comment on the accuracy and stability. Note that Higham's experiments are run in double precision and compared to "exact" values from Matlab's 50 digit symbolic arithmetic toolbox so you will not see exactly the same behavior.

Task 5

For function $f_4(x)$ perform the following subtasks:

1. For the uniform and Chebyshev meshes with a range of values of n and number of points $n+1$, assess the accuracy and stability of the single precision codes using the appropriate bounds from the notes and literature on the class webpage and the values generated in determining the conditioning in the subtask above. This should be done for

   - Barycentric form 2 of the polynomial
   - Newton form with the mesh points in increasing order, decreasing order and satisfying the Leja ordering conditions.

   You should provide plots similar to those in Higham's paper for this function on the uniform mesh and the Chebyshev points of the first kind. The other examples should be summarized to comment on the accuracy and stability. Note that Higham's experiments are run in double precision and compared to "exact" values from Matlab's 50 digit symbolic arithmetic toolbox so you will not see exactly the same behavior.

2. Investigate the convergence (or lack thereof) to $f_4(x)$ as n increases for the Barycentric Form 2 with the uniform points and Chebyshev points of the first and second kind. For the convergent mesh family, empirically determine how large n must be to achieve various levels approximation as measured by $\|f_4(x) - p_n(x)\|_\infty$. Is there a threshold below which you cannot go in your observations?