Math 781: Project # 3

Polynomial Interpolation Using Newton’s Divided Differences

(due: Tuesday, 11/4/14)

Implement the polynomial interpolation using Newton’s divided differences. See the algorithms at the end of this assignment. It is strongly recommended that the algorithms Divdif($d, x, n$) and Interp($d, x, n, t, p$) be programmed in (separate) functions (or subroutines).

Consider the famous Carl Runge example

\[ f(x) = \frac{1}{1 + x^2}, \quad -5 \leq x \leq 5 \]

Use two sets of nodes

Set I: \( x_i = -5 + i \, h, \quad i = 0, 1, \ldots, n, \quad h = \frac{10}{n} \)

Set II: \( x_i = -5 \cos \left( \frac{i\pi}{n} \right), \quad i = 0, 1, \ldots, n \)

The first set of nodes are equidistant while the second set, often referred to as the Gauss-Lobatto Chebyshev points, are denser at the ends than in the middle.

Denote by \( p_n(x) \) the interpolation polynomial satisfying the conditions

\[ p_n(x_i) = f(x_i), \quad i = 0, 1, \ldots, n \]

Define the maximum error as

\[ E_n = \max_{i=0,\ldots,n-1} \left| p_n \left( \frac{x_i + x_{i+1}}{2} \right) - f \left( \frac{x_i + x_{i+1}}{2} \right) \right|. \]

Use your program to compute the error for various values of \( n \), say \( n = 5, 10, 20, 40, \) and \( 80 \), for the two sets of nodes. What can you conclude based on the results you have obtained? (Hints for analysis: convergence and convergence rate; graph or table for the error \( E_n \) as function of \( n \).)

Turn in your report. See Project # 1 for the requirements for the format of project reports.

Algorithm Divdif($d, x, n$) (for computing the divided differences)

1. Remark: On entrance, \( d \) and \( x \) are vectors with \( f(x_i) \) and \( x_i, \quad i = 0, \ldots, n \).

   On exit, \( d_i \) contains \( f[x_0, \ldots, x_i] \).

2. Do through Step 4 for \( i = 1, 2, \ldots, n \)
3. Do through Step 4 for $j = n, n - 1, ..., i$

4. $d_j := (d_j - d_{j-1})/(x_j - x_{j-i})$

5. Exit from the algorithm.

**Algorithm** Interp($d, x, n, t, p$) (for computing $p_n(t)$ at point $t$)

1. Remark: On entrance, $d$ and $x$ are vectors containing $f[x_0, ..., x_i]$ and $x_i, i = 0, ..., n$.
   On exit, $p$ will contain the value $p_n(t)$.

2. $p := d_n$

3. Do through Step 4 for $i = n - 1, n - 2, ..., 0$

4. $p := d_i + (t - x_i)p$

5. Exit the algorithm.