Math 104A Homework #3 *

Instructor: Xu Ynag

General Instructions: Please write your homework papers neatly. You need to turn in both your code and descriptions on Canvas with the appropriate form that the TA requires. Write your own code individually. Do not copy codes!

1. (a) Let $f \in C^2[x_0, x_1]$ and P_1 its interpolation linear polynomial at x_0 and x_1. Prove that

$$\|f - P_1\|_\infty \leq \frac{1}{8}(x_1 - x_0)^2 M_2, \tag{1}$$

where $|f''(x)| \leq M_2$ for all $x \in [x_0, x_1]$ and $\|f - P_1\|_\infty = \max_{x \in [x_0, x_1]} |f(x) - P_1(x)|$.

(b) Let $P_1(x)$ be the linear polynomial that interpolates $f(x) = \sin x$ at $x_0 = 0$ and $x_1 = \pi/2$. Using (a) find a bound for the maximum error $\|f - P_1\|_\infty$ and compare this bound with the actual error at $x = \pi/4$.

2. (a) Equating the leading coefficient of in the Lagrange form of the interpolation polynomial $P_n(x)$ with that of the Newton's form deduce that

$$f[x_0, x_1, ..., x_n] = \sum_{j=0}^{n} \frac{f(x_j)}{\prod\limits_{\substack{k=0 \\ k \neq j}}^{n} (x_j - x_k)}. \tag{2}$$

(b) Use (a) to conclude that divided differences are symmetric functions of their arguments, i.e. any permutation of $x_0, x_1, ..., x_n$ leaves the corresponding divided difference unchanged.

3. In Newton's form of the interpolation polynomial we need to compute the coefficients, $c_0 = f[x_0]$, $c_1 = f[x_0, x_1]$, ..., $c_n = f[x_0, x_1, ..., x_n]$. In the table of divided differences we proceed column by column and the needed coefficients are in the uppermost diagonal. A simple 1D array, c of size n + 1, can be used to store and compute these values. We just have to compute them from bottom to top to avoid losing values we have already computed. The following pseudocode does precisely this:

for $j = 0, 1, ..., n$ **do**

$c_j = f_j$
end for
for $k = 1, ..., n$ **do**
  for $j = n, n-1, ..., k$ **do**
    $c_j = (c_j - c_{j-1})/(x_j - x_{j-k})$
  end for
end for

The evaluation of the interpolation polynomial in Newton's form can be then done with the Horner-like scheme:

$p = c_n$
for $j = n, n-1, ..., 0$ **do**
  $p = c_j + (x - x_j)p$
end for

(a) Write computer codes to compute the coefficients $c_0, c_1, ..., c_n$ and to evaluate the corresponding interpolation polynomial at an arbitrary point x. Test your codes and turn in a run of your test.

(b) Consider the function $f(x) = e^{-x^2}$ for $x \in [-1, 1]$ and the nodes $x_j = -1 + j(2/10)$, $j = 0, 1, ..., 10$. Use your code in (a) to evaluate $P_{10}(x)$ at the points $\bar{x}_j = -1 + j(2/100)$, $j = 0, 1, ..., 100$ and plot the error $f(x) - P_{10}(x)$.

4. *Inverse Interpolation.* Suppose that we want to solve the equation $f(x) = 0$, for some function f which has an inverse f^{-1}. If we have two approximations x_0 and x_1 of a zero $\bar{x}$ of f then we can use interpolation to find a better approximation, $\bar{x} \approx f^{-1}(0)$, as follows. Let $y_0 = f(x_0)$ and $y_1 = f(x_1)$.

Table 1

$y_j = f(x_j)$	x_j	
y_0	x_0	
y_1	x_1	$f^{-1}[y_0, y_1]$

and $P_1(0) = x_0 + f^{-1}[y_0, y_1](0 - y_0) = x_0 - y_0 f^{-1}[y_0, y_1]$. We could now define $x_2 = P_1(0)$, evaluate f at this point to get $y_2 = f(x_2)$, and then add one more row to our table 1 to get $f^{-1}[y_0, y_1, y_2]$. Once this is computed we can evaluate $P_2(0)$ to get an improved approximation $\bar{x}$, etc. Let $f(x) = x - e^{-x}$ using the values $f(0.5) = -0.106530659712633$ and $f(0.6) = 0.051188363905973$ find an approximate value for the zero $\bar{x}$ of f by evaluating $P_1(0)$.

5. Obtain the Hermite interpolation polynomial corresponding to the data $f(0) = 0$, $f'(0) = 0$, $f(1) = 2$, $f'(1) = 3$.