

**Math 151A Homework #4** – due Wednesday 11/01, in class

Show all your work!

**1. Error term for Taylor's theorem** (section 3.1, problem 31)

In class we proved Theorem 3.3, which stated the error term for the Lagrange interpolating polynomial. Using the same procedure, prove Taylor's theorem (Theorem 1.14). You will want to use the function

$$g(t) = f(t) - P(t) - [f(x) - P(x)] \cdot \frac{(t - x_0)^{n+1}}{(x - x_0)^{n+1}}$$

where  $P(x)$  is the  $n$ th Taylor polynomial.

**Answer:**

Let  $f \in C^n[a, b]$ ,  $f^{(n+1)}$  exists on  $[a, b]$  and  $x_0 \in [a, b]$ . Let  $x$  be some point in  $[a, b]$  and

$$P(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f^{(2)}(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}}{n!}(x - x_0)^n.$$

Then

$$\begin{aligned} g(x) &= f(x) - P(x) - [f(x) - P(x)] \cdot \frac{(x - x_0)^{n+1}}{(x - x_0)^{n+1}} \\ &= f(x) - P(x) - [f(x) - P(x)] = 0 \end{aligned}$$

and

$$\begin{aligned} g(x_0) &= f(x_0) - P(x_0) - [f(x) - P(x)] \cdot \frac{(x_0 - x_0)^{n+1}}{(x - x_0)^{n+1}} \\ &= f(x_0) - P(x_0) = 0 \end{aligned}$$

because  $f(x_0) = P(x_0)$ . Moreover, notice that  $f^{(k)}(x_0) = P^{(k)}(x_0)$  for all  $0 \leq k \leq n + 1$ . Since  $g$  is a continuous, differentiable function on  $[a, b]$  and  $g$  is zero at the two points  $x, x_0 \in [a, b]$ , then Rolle's theorem says that a  $\xi_1 \in [x, x_0]$  exists with  $g'(\xi_1) = 0$ .

Now we have that  $g'(\xi_1) = 0$  and  $g'(x_0) = 0$ , so by Rolle's theorem, there exists a  $\xi_2 \in [\xi_1, x_0]$  with  $g^{(2)}(\xi_2) = 0$ .

And so on, until  $g^{(n)}(\xi_n) = g^{(n)}(x_0) = 0$  and Rolle's theorem tells us that there exists a  $\xi_{n+1} \in [\xi_n, x_0]$  with  $g^{(n+1)}(\xi_{n+1}) = 0$ . Since  $P$  is an  $n$ -degree polynomial,  $P^{(n+1)}$  is identically zero. And so

$$g^{(n+1)}(\xi_{n+1}) = 0 = f^{(n+1)}(\xi_{n+1}) - [f(x) - P(x)] \cdot \frac{(n+1)!}{(x - x_0)^{n+1}}$$

and

$$f(x) = P(x) + \frac{f^{(n+1)}(\xi_{n+1})}{(n+1)!}(x - x_0)^{n+1}$$

which is what Taylor's theorem states.

**2. Lagrange interpolation** (section 3.1, problems 1a & 3)

Given the function  $f(x) = \cos x$ , use the points  $x_0 = 0$ ,  $x_1 = 0.6$  and  $x_2 = 0.9$  to do the following:

- Construct Lagrange polynomials of degree at most one and at most two to approximate  $f(x)$ .

**Answer:**

For a first degree interpolating polynomial we need the first degree Lagrange polynomials. We will use  $x_0$  and  $x_1$  to construct them:

$$L_{1,0} = \frac{(x-x_1)}{(x_0-x_1)} = \frac{(x-0.6)}{(-0.6)} = -1.666667x + 1$$

$$L_{1,1} = \frac{(x-x_0)}{(x_1-x_0)} = \frac{(x)}{(0.6)} = 1.666667x$$

which yields an interpolating polynomial of

$$P_1(x) = L_{1,0}f(x_0) + L_{1,1}f(x_1)$$

$$= (-1.666667x + 1)\cos(0) + (1.666667x)\cos(0.6)$$

$$= -0.2911073x + 1$$

For second degree, we use the three second degree Lagrange polynomials

$$L_{2,0} = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-0.6)(x-0.9)}{(0-0.6)(0-0.9)} = \frac{x^2 - 1.5x + 0.54}{0.54}$$

$$L_{2,1} = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-0)(x-0.9)}{(0.6-0)(0.6-0.9)} = \frac{x^2 - 0.9x}{-0.18}$$

$$L_{2,2} = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-0)(x-0.6)}{(0.9-0)(0.9-0.6)} = \frac{x^2 - 0.6x}{0.27}$$

and get

$$P_2(x) = L_{2,0}f(x_0) + L_{2,1}f(x_1) + L_{2,2}f(x_2)$$

$$= \frac{x^2 - 1.5x + 0.54}{0.54}\cos(0) + \frac{x^2 - 0.9x}{-0.18}\cos(0.6) + \frac{x^2 - 0.6x}{0.27}\cos(0.9)$$

$$= (-0.43108687)x^2 + (-0.03245519)x + 1.$$

b. Use these polynomial approximations to compute the value of  $\cos(0.45)$  and the absolute error.

**Answer:**

$$P_1(0.45) = 0.8690017, P_2(0.45) = 0.8981007, \text{ and } \cos(0.45) = 0.900447.$$

$$\text{So } |\cos(0.45) - P_1(0.45)| = 0.0314454 \text{ and } |\cos(0.45) - P_2(0.45)| = 0.002347.$$

c. Use theorem 3.3 to calculate an error bound for these approximations. Do the answers from part (b) obey the error bounds?

**Answer:**

Theorem 3.3 tells us that the absolute error is given by

$$|f(x) - P(x)| = \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{i=0}^n (x-x_i).$$

So then,

$$|f(x) - P_1(x)| = \left| \frac{f^{(2)}(\xi)}{2} (0.45-0)(0.45-0.6) \right|$$

$$\leq \frac{\cos(0)}{2} (0.0675) = 0.03375$$

and

$$|f(x) - P_2(x)| = \left| \frac{f^{(3)}(\xi)}{6} (0.45 - 0)(0.45 - 0.6)(0.45 - 0.9) \right|$$

$$\leq \frac{\sin(0.9)}{6} (0.03075) = 0.00397$$

The absolute errors found in part (b) are smaller than the theoretical bounds computed above.

### 3. Neville's method [computational] (section 3.1, problem 11)

Use Neville's method to approximate  $\sqrt{3}$  with the following functions and values

**Answer:**

Here is Matlab/Octave code for Neville's method:

```
%-----
% Neville's method for polynomial interpolation
% must put x0,x1,...,xn in vector called "nodes"
% and f(x0),f(x1),... in vector called "f"
% and value to find approx at called "x"
%-----

% create matrix to store up Q_{i,j}
n = length(nodes);
Q = zeros(n);

% fill up first column
for i = 1:n
    Q(i,1) = f(i);
end

% now fill up row by row
for i = 2:n
    for j = 2:i
        Q(i,j) = ( (x - nodes(i-j+1))*Q(i,j-1) - (x-nodes(i))*Q(i-1,j-1) ) ...
            / ( nodes(i)-nodes(i-j+1) );
    end
end

% the diagonal elements of Q are the successive approximations
diag(Q)
```

- a.  $f(x) = 3^x$  and the values  $x_0 = -2, x_1 = -1, x_2 = 0, x_3 = 1$  and  $x_4 = 2$ .

**Answer:**

To approximate  $\sqrt{3}$ , we use Neville's method to approximate  $f(x) = 3^x$  at  $x = 0.5$ :

```
octave:1> nodes=[-2,-1,0,1,2]; f=3.^nodes; x=0.5; neville
```

```
ans =
    0.11111
    0.66667
    1.50000
    1.77778
    1.70833
```

b.  $f(x) = \sqrt{x}$  and the values  $x_0 = 0, x_1 = 1, x_2 = 2, x_3 = 4$  and  $x_4 = 5$

**Answer:**

To approximate  $\sqrt{3}$ , we use Neville's method to approximate  $f(x) = \sqrt{x}$  at  $x = 3$ :

```
octave:2> nodes=[0,1,2,4,5]; f=sqrt(nodes); x=3; neville
```

```
ans =
    0.00000
    3.00000
    1.24264
    1.62132
    1.69061
```

c. Compare the accuracy of the approximation in parts (a) and (b).

**Answer:**

We see that part (a) gives an absolute error of  $|\sqrt{3} - 1.70833| = 0.023721$ ; whereas part (b) gives an absolute error of 0.041441, which is larger.

#### 4. Newton's divided difference [computational] (section 3.2, problem 7a)

Use Algorithm 3.2 to construct the interpolating polynomial of degree three for the unequally spaced points given in the following table:

x	f(x)
-0.1	5.300
0.0	2.000
0.2	3.190
0.3	1.000

**Answer:**

Newton's divided difference code (ndd.m):

```
%-----
% Newton divided difference for polynomial interpolation
% must put x0,x1,...,xn in vector called "nodes"
% and f(x0),f(x1),... in vector called "f"
%-----

% create matrix to store up F_{i,j}
n = length(nodes);
F = zeros(n);

% fill up first column
for i = 1:n
    F(i,1) = f(i);
end

% now fill up row by row
for i = 2:n
    for j = 2:i
        F(i,j) = ( F(i,j-1) - F(i-1,j-1) ) / ( nodes(i) - nodes(i-j+1) );
    end
end
```

end

```
% print out diagonal elements
```

```
diag(F)
```

generates the coefficients:

```
octave:3> nodes=[-0.1,0.0,0.2,0.3]; f=[5.3,2.0,3.19,1.0]; nnd
```

```
ans =
```

```
5.3000
```

```
-33.0000
```

```
129.8333
```

```
-556.6667
```

which yields the interpolating polynomial

$$P(x) = 5.3 - 33.0(x + 0.1) + 129.8333(x + 0.1)(x) - 556.6667(x + 0.1)(x)(x - 0.2)$$

