2021 Spring MAS 365: Homework 6

posted on May 6; due by May 13

1. [10+10 points]

   (a) Construct the second Lagrange interpolating polynomial for the function

   $$f(x) = x^{-3}, \quad \text{and the nodes } x_0 = 1, x_1 = 2, x_2 = 3.$$

   You don't have to write the polynomial in a compact form.

   (b) Find a bound for the corresponding absolute error on the interval $[x_0, x_2]$.

Solution:

(a) We have

$$\begin{aligned}
L_0(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-2)(x-3)}{(1-2)(1-3)} = \frac{1}{2}(x-2)(x-3)\\
L_1(x) &= \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-1)(x-3)}{(2-1)(2-3)} = -(x-1)(x-3)\\
L_2(x) &= \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-1)(x-2)}{(3-1)(3-2)} = \frac{1}{2}(x-1)(x-2).
\end{aligned}$$

Since $f(x_0) = 1$, $f(x_1) = \frac{1}{8}$, $f(x_2) = \frac{1}{27}$, we have

$$P(x) = \sum_{k=0}^{2} f(x_k)L_k(x) = \frac{1}{2}(x-2)(x-3) - \frac{1}{8}(x-1)(x-3) + \frac{1}{54}(x-1)(x-2).$$

(b) Since $f'''(x) = -60x^{-6}$, the second Lagrange polynomial has the error form

$$\frac{f'''(\xi(x))}{3!}(x-x_0)(x-x_1)(x-x_2) = \frac{-60(\xi(x))^{-6}}{6}(x-1)(x-2)(x-3)$$

for $\xi(x)$ in $(1,3)$. The maximum value of $(\xi(x))^{-6}$ on the interval is 1. We now need to determine the maximum value on this interval of the absolute value of the polynomial

$$g(x) = (x-1)(x-2)(x-3) = x^3 - 6x^2 + 11x - 6.$$

Since

$$g'(x) = 3x^2 - 12x + 11,$$

the critical points occur at

$$x = 2 + \frac{1}{\sqrt{3}} \text{ with } g\left(2+\frac{1}{\sqrt{3}}\right) = -\frac{2}{3\sqrt{3}} \quad \text{and} \quad x = 2 - \frac{1}{\sqrt{3}} \text{ with } g\left(2-\frac{1}{\sqrt{3}}\right) = \frac{2}{3\sqrt{3}}.$$

Hence, the maximum error is

$$\left|\frac{f'''(\xi(x))}{3!}(x-x_0)(x-x_1)(x-x_2)\right| \le 10 \cdot \frac{2}{3\sqrt{3}} = \frac{20}{3\sqrt{3}} \approx 3.8490.$$

2. [10 points] Prove Taylor's Theorem 1.14 in the textbook by following the procedure in the proof of Theorem 3.3 in the textbook. [Hint: Let $g(t) = f(t) - P(t) - [f(x) - P(x)]\frac{(t-x_0)^{n+1}}{(x-x_0)^{n+1}}$, where P is the nth Taylor polynomial, and use the Generalized Rolle's Theorem and $g(x_0) = g'(x_0) = g''(x_0) = \cdots = g^{(n)}(x_0) = 0$.]

Solution: Since $g(x) = g(x_0) = 0$, there exists ξ_1 between x_0 and x, for which $g'(\xi_1) = 0$. Then, since $g'(x_0) = 0$, there exists ξ_2 between x_0 and ξ_1, for which $g''(\xi_2) = 0$. By continuing this process, we can show that there exists ξ_{n+1} between x_0 and ξ_n, for which $g^{(n+1)}(\xi_{n+1}) = 0$. Then, since $P(x)$ is a polynomial of degree at most n, we have

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

and this concludes the proof. □

3. [10 points] Neville's method is used to approximate $f(0.4)$, giving the following table.

$x_0 = 0$	$P_0 = 1$			
$x_1 = 0.25$	$P_1 = 2$	$P_{0,1} = 2.6$		
$x_2 = 0.5$	P_2	$P_{1,2}$	$P_{0,1,2}$	
$x_3 = 0.75$	$P_3 = 8$	$P_{2,3} = 2.4$	$P_{1,2,3} = 2.96$	$P_{0,1,2,3} = 3.016$

Determine $P_2 = f(0.5)$.

Solution: We have

$$2.4 = P_{2,3} = \frac{(x-x_2)P_3 - (x-x_3)P_2}{x_3 - x_2} = \frac{(0.4-0.5)8 - (0.4-0.75)P_2}{0.75-0.5},$$

so

$$P_2 = \frac{0.6+0.8}{0.35} = 4.$$

4. [10 points] Show that $H_{2n+1}(x)$ is the unique polynomial of least degree agreeing with f and f' at $x_0, \ldots, x_n$. Assume that $P(x)$ is another such polynomial and consider $D(x) = H_{2n+1}(x) - P(x)$ and $D'(x)$ at $x = x_0, x_1, \ldots, x_n$.

Solution: Suppose $P(x)$ is another polynomial of degree at most $2n+1$, agreeing with f and f' at $x_0, \ldots, x_n$. Let $D(x) = H_{2n+1}(x) - P(x)$. Then $D(x)$ is a polynomial of degree at most $2n+1$ with $D(x_i) = 0$ and $D'(x_i) = 0$ for $i = 0, \ldots, n$. This is true only if $D(x) = q(x)\prod_{i=0}^{n}(x - x_i)^2$ for some $q(x)$. Hence, if $q(x) \neq 0$, $D(x)$ must be of degree at least $2n+2$, which is a contradiction. So, we need $q(x) = 0$, which implies $D(x) = 0$ and thus $P(x) = H_{2n+1}(x)$.

5. [10+5 points] Let $(x_0, y_0) = (0, 0)$ and $(x_1, y_1) = (5, 2)$ be the endpoints of a curve, and let $(1, 1)$ and $(6, 3)$ be the given guidepoints, respectively.

   (a) Construct a parametric cubic Hermite approximations $(x(t), y(t))$ to the curve.

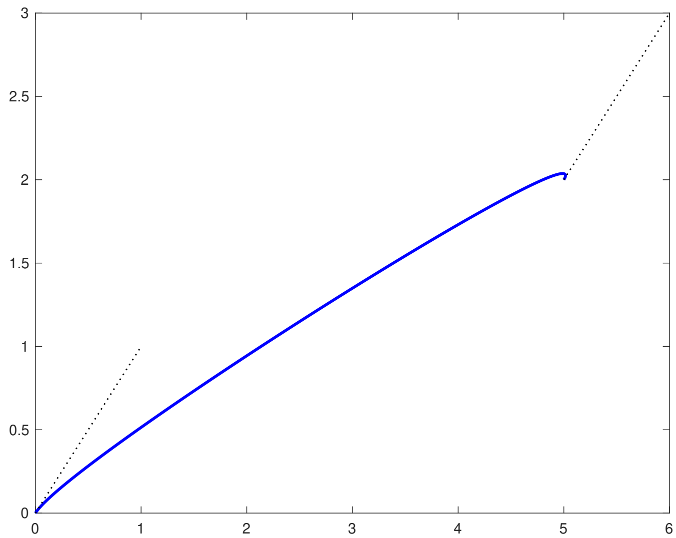
   (b) Draw a graph of the approximation (possibly by MATLAB).

Solution:

(a) Since $(\alpha_0, \beta_0) = (1, 1)$ and $(\alpha_1, \beta_1) = (-1, -1)$, we have

$$x(t) = [2(0-5) + (1-1)]t^3 + [3(5-0) - (-1+2)]t^2 + t + 0 = -10t^3 + 14t^2 + t$$
$$y(t) = [2(0-2) + (1-1)]t^3 + [3(2-0) - (-1+2)]t^2 + t + 0 = -4t^3 + 5t^2 + t$$

(b) The graph of the approximation $(x(t), y(t))$ is sufficient for the answer, and the graph regarding the guidepoint is added for illustration.



6. [10+10+10 points]

   (a) Implement the Newton divided difference formula via MATLAB grader.

   (b) Implement the divided difference formula for Hermite Polynomials via MATLAB grader.

   (c) Implement the cubic spline interpolations via MATLAB grader.

   Solution:

   (a)
```
function [a b] = newton_divided_diff(x, f)
    n = length(x)-1;
    F = zeros(n+1,n+1);

    F(:,1) = f;
    for i=1:n
    for j=1:i
       F(i+1,j+1) = (F(i+1,j) - F(i,j))/(x(i+1) - x(i-j+1));
    end
    end

    a = diag(F);
    b = F(end,:)';
end
```

   (b)
```
function [a b] = divided_diff_for_Hermite(x, f, g)
    n = length(x)-1;
    z = zeros(2*n+2,1);
    Q = zeros(2*n+2,2*n+2);

    z(1:2:end) = x;
    z(2:2:end) = x;
```

```
    Q(1:2:end,1) = f;
    Q(2:2:end,1) = f;
    Q(2:2:end,2) = g;
    for i=2:2:2*n+1
        Q(i+1,2) = (Q(i+1,1) - Q(i,1))/(z(i+1) - z(i));
    end
    for i=2:2*n+1
    for j=2:i
       Q(i+1,j+1) = (Q(i+1,j) - Q(i,j))/(z(i+1) - z(i-j+1));
    end
    end

    a = diag(Q);
    b = Q(end,:)';
end
```

(c)

```
function [a b c d] = cubic_spline(x, f, opt, g0, gn)
    n = length(x)-1;
    a = zeros(n+1,1); b = zeros(n,1); c = zeros(n+1,1); d = zeros(n,1);
    h = zeros(n,1); l = zeros(n+1,1); mu = zeros(n,1); z = zeros(n+1,1);

    a = f;
    for i=0:n-1
        h(i+1) = x(i+2) - x(i+1);
    end
    if opt
       alp(1) = 3*(a(2) - a(1))/h(1) - 3*g0;
       alp(n+1) = 3*gn - 3*(a(n+1) - a(n))/h(n);
    end
    for i=1:n-1
        alp(i+1) = 3/h(i+1)*(a(i+2) - a(i+1)) - 3/h(i)*(a(i+1) - a(i));
    end

    if opt == 0
        l(1) = 1; mu(1) = 0; z(1) = 0;
    else
        l(1) = 2*h(1); mu(1) = 1/2; z(1) = alp(1)/l(1);
    end
    for i=1:n-1
        l(i+1) = 2*(x(i+2) - x(i)) - h(i)*mu(i);
        mu(i+1) = h(i+1)/l(i+1);
        z(i+1) = (alp(i+1) - h(i)*z(i))/l(i+1);
    end
    if opt == 0
        l(n+1) = 1; z(n+1) = 0; c(n+1) = 0;
    else
        l(n+1) = h(n)*(2-mu(n));
        z(n+1) = (alp(n+1) - h(n)*z(n))/l(n+1); c(n+1) = z(n+1);
    end

    for j=n-1:-1:0
```

```
        c(j+1) = z(j+1) - mu(j+1)*c(j+2);
        b(j+1) = (a(j+2) - a(j+1))/h(j+1) - h(j+1)*(c(j+2)+2*c(j+1))/3;
        d(j+1) = (c(j+2) - c(j+1))/3/h(j+1);
    end

    a = a(1:end-1);
    c = c(1:end-1);
end
```