Math128aHw5

Trustin Nguyen

October 9, 2024

Exercise Set 3.4

Exercise 6: Let $f(x) = 3xe^x - e^{2x}$.

(a) Approximate $f(1.03)$ by the Hermite interpolating polynomial of degree at most three using $x_0 = 1$ and $x_1 = 1.05$. Compare the actual error to the error bound.

Answer. I got $P(1.03) = 0.80932385725980970914150702810 73$ using the divided difference method. The actual value is 0.809323618901705674426271697866 69. The absolute error is 0.00000023835810403471523533024060767165

Here is my code:

```
function f = Hermite(x, y, dy)
    sz = size(x, 1);

    p = zeros(2 * sz, 2*sz + 1);
    p(1:2:end, 1) = x;
    p(2:2:end, 1) = x;
    p(1:2:end, 2) = y;
    p(2:2:end, 2) = y;
    p(1:2:end, 3) = dy;
    p(2:2:end, 3) = dy;
    p(1, 3) = 0.0;

    dup_x = p(:, 1);
    for i = 1:sz-1
        p(2*i+1, 3) = (p(2*i+1, 2) - p(2*i, 2)) / (dup_x(2*i+1) - dup_x(2*i));
    end

    for i = 3:2*sz
        for j = 3:i
            p(i, j+1) = (p(i, j) - p(i-1, j)) / (dup_x(i) - dup_x(i-j+1));
        end
    end

    f = zeros(2*sz, 1);
    for i = 1:2*sz
        f(i) = p(i, i+1);
    end

end

function f = forwardPoly(x, c, d)
    syms s;
```

```
    f = c(d);
    new_x = zeros(2*size(x, 1), 1);
    new_x(1:2:end) = x;
    new_x(2:2:end) = x;
    x = new_x;
    for i = 1:d+1
        f = f * (s - x(d-i+2));
        f = f + c(d-i+2);
    end
end

syms s
f(s) = 3*s*exp(s) - exp(2*s);
df = diff(f, s);

x = [1; 1.05];
y = double(f(x));
dy = double(df(x));
coeff = Hermite(x, y, dy);
P = matlabFunction(forwardPoly(x, coeff, 3));
res = vpa(P(1.03))
actual = vpa(f(1.03))
err = abs(res - actual)
```

(b) Repeat (a) with the Hermite interpolating polynomial of degree at most five using $x_0 = 1$, $x_1 = 1.05$, and $x_2 = 1.07$.

Answer. I got $P(1.03) = 0.809323617244237070167400815989982$ with the actual value being $f(1.03) = 0.80932361890170567442627169786669$. The error is therefore 0.0000000016574686042588708818768687171581

Here is my code:

```
function f = Hermite(x, y, dy)
    sz = size(x, 1);

    p = zeros(2 * sz, 2*sz + 1);
    p(1:2:end, 1) = x;
    p(2:2:end, 1) = x;
    p(1:2:end, 2) = y;
    p(2:2:end, 2) = y;
    p(1:2:end, 3) = dy;
    p(2:2:end, 3) = dy;
    p(1, 3) = 0.0;

    dup_x = p(:, 1);
    for i = 1:sz-1
        p(2*i+1, 3) = (p(2*i+1, 2) - p(2*i, 2)) / (dup_x(2*i+1) - dup_x(2*i));
    end

    for i = 3:2*sz
        for j = 3:i
            p(i, j+1) = (p(i, j) - p(i-1, j)) / (dup_x(i) - dup_x(i-j+1));
        end
    end

    f = zeros(2*sz, 1);
    for i = 1:2*sz
```

```
            f(i) = p(i, i+1);
        end

    end

    function f = forwardPoly(x, c, d)
        syms s;
        f = c(d);
        new_x = zeros(2*size(x, 1), 1);
        new_x(1:2:end) = x;
        new_x(2:2:end) = x;
        x = new_x;
        for i = 1:d+1
            f = f * (s - x(d-i+2));
            f = f + c(d-i+2);
        end
    end

    syms s
    f(s) = 3*s*exp(s) - exp(2*s);
    df = diff(f, s);

    x = [1; 1.05; 1.07];
    y = double(f(x));
    dy = double(df(x));
    coeff = Hermite(x, y, dy);
    P = matlabFunction(forwardPoly(x, coeff, 5));
    res = vpa(P(1.03))
    actual = vpa(f(1.03))
    err = abs(res - actual)
```

Exercise 7: The following table lists data for the function described by $f(x) = e^{0.1x^2}$. Approximate $f(1.25)$ by using $H_5(1.25)$ and $H_3(1.25)$, where H_5 uses the nodes $x_0 = 1, x_1 = 2$, and $x_2 = 3$ and H_3 uses the nodes $\overline{x_0} = 1$ and $\overline{x_1} = 1.5$. Find error bounds for these approximations.

x	$f(x) = e^{0.1x^2}$	$f'(x) = 0.2xe^{0.1x^2}$
$x_0 = \overline{x_0} = 1$	1.105170918	0.2210341836
$\overline{x_1} = 1.5$	1.252322716	0.3756968148
$x_1 = 2$	1.491824698	0.5967298792
$x_2 = 3$	2.459603111	1.475761867

Answer. Using two nodes, I got $P(1.25) = 1.1696528199671771819367904754472$ while the true value is $f(1.25) = 1.1691184461695044022981846915119$. The error is then 0.00053437379767277963860578393530507.

Using three nodes, I got $P(1.25) = 1.1709556512384045046104574794299$ with the true value being $f(1.25) = 1.1691184461695044022981846915119$. Then the error is 0.0018372050689001023

We see that this time, using two nodes was better than using three, because the two nodes were closer to where we wanted to approximate f at.

Here is my code:

```
function f = Hermite(x, y, dy)
    sz = size(x, 1);

    p = zeros(2 * sz, 2*sz + 1);
```

```
    p(1:2:end, 1) = x;
    p(2:2:end, 1) = x;
    p(1:2:end, 2) = y;
    p(2:2:end, 2) = y;
    p(1:2:end, 3) = dy;
    p(2:2:end, 3) = dy;
    p(1, 3) = 0.0;

    dup_x = p(:, 1);
    for i = 1:sz-1
        p(2*i+1, 3) = (p(2*i+1, 2) - p(2*i, 2)) / (dup_x(2*i+1) - dup_x(2*i));
    end

    for i = 3:2*sz
        for j = 3:i
            p(i, j+1) = (p(i, j) - p(i-1, j)) / (dup_x(i) - dup_x(i-j+1));
        end
    end

    f = zeros(2*sz, 1);
    for i = 1:2*sz
        f(i) = p(i, i+1);
    end

end

function f = forwardPoly(x, c, d)
    syms s;
    f = c(d);
    new_x = zeros(2*size(x, 1), 1);
    new_x(1:2:end) = x;
    new_x(2:2:end) = x;
    x = new_x;
    for i = 1:d+1
        f = f * (s - x(d-i+2));
        f = f + c(d-i+2);
    end
end

syms s
f(s) = exp(0.1*s^2);
df = diff(f, s);

x = [1; 1.5];
y = double(f(x));
dy = double(df(x));
coeff = Hermite(x, y, dy);
P = matlabFunction(forwardPoly(x, coeff, 3));
res = vpa(P(1.25))
actual = vpa(f(1.25))
err = abs(res - actual)

syms s
f(s) = exp(0.1*s^2);
df = diff(f, s);

x = [1; 2; 3];
```

```
y = double(f(x));
dy = double(df(x));
coeff = Hermite(x, y, dy);
P = matlabFunction(forwardPoly(x, coeff, 5));
res = vpa(P(1.25))
actual = vpa(f(1.25))
err = abs(res - actual)
```

Exercise 10: Let $z_0 = x_0, z_2 = x_1$, and $z_3 = x_1$. Form the following divided-difference table.

$$
\begin{array}{lllll}
z_0 = x_0 & f[z_0] = f(x_0) & & & \\
 & & f[z_0, z_1] = f'(x_0) & & \\
z_1 = x_0 & f[z_1] = f(x_0) & & f[z_0, z_1, z_2] & \\
 & & f[z_1, z_2] & & f[z_0, z_1, z_2, z_3] \\
z_2 = x_1 & f[z_2] = f(x_1) & & f[z_1, z_2, z_3] & \\
 & & f[z_2, z_3] = f'(x_1) & & \\
z_3 = x_1 & f[z_3] = f(x_1) & & &
\end{array}
$$

Show that the cubic Hermite polynomial $H_3(x)$ can also be written as $f[z_0] + f[z_0, z_1](x - x_0) + f[z_0, z_1, z_2](x - x_0)^2 + f[z_0, z_1, z_2, z_3](x - x_0)^2(x - x_1)$

Answer. The Hermite polynomial is given by:

$$H_{2n+1}(x) = f[z_0] + \sum_{k=1}^{2n+1} f[z_0, \dots, z_k](x - z_0)(x - z_1) \cdots (x - z_{k-1})$$

There are two nodes so $n = 1$. Then we get:

$$H_3(x) = f[z_0] + f[z_0, z_1](x - z_0) + f[z_0, z_1, z_2](x - z_0)(x - z_1) + f[z_0, z_1, z_2, z_3](x - z_0)(x - z_1)(x - z_2)$$

Substitute in $z_0, z_1, z_2, z_3 = x_0, x_0, x_1, x_1$ to get:

$$H_3(x) = f[z_0] + f[z_0, z_1](x - x_0) + f[z_0, z_1, z_2](x - x_0)^2 + f[z_0, z_1, z_2, z_3](x - x_0)^2(x - x_1)$$

Exercise Set 3.5

Exercise 1: Determine the natural cubic spline S that interpolates the data $f(0) = 0$, $f(1) = 1$, ad $f(2) = 2$.

Answer. For a natural cubic spline, we require the conditions:

(a) $S_i(x_{i+1}) = S_{i+1}(x_{i+1})$

(b) $S'_i(x_{i+1}) = S'_{i+1}(x_{i+1})$

(c) $S''_i(x_{i+1}) = S''_{i+1}(x_{i+1})$.

Then we have

$$S_0(x) = 0 + b_0(x-0) + c_0(x-0)^2 + d_0(x-0)^3$$
$$S_1(x) = 1 + b_1(x-1) + c_1(x-1)^2 + d_1(x-1)^3$$

By condition (a):

$$S_0(1) = b_0 + c_0 + d_0 = 1$$
$$S_1(2) = 1 + b_1 + c_1 + d_1 = 2$$

By condition (b):

$$S'_0(1) = b_0 + 2c_0 + 3d_0$$
$$S'_1(1) = b_1$$

By condition (c):

$$S''_0(1) = 2c_0 + 6d_0$$
$$S''_1(1) = 2c_1$$

are equal. Because it is a natural spline,

$$S''_0(0) = 2c_0 = 0$$
$$S''_1(2) = 2c_1 + 6d_1 = 0$$

By condition (c), $2c_1 = 6d_0$, and from the natural boundary condition:

$$0 = d_0 + d_1, d_0 = -d_1$$

From condition (a), $b_0 + d_0 = 1$, and from (b),

$$S'_0(1) = 2d_0 + 1 = b_1 = S'_1(1)$$

so

$$2d_0 + 1 = b_1$$

From condition (a),

$$S_1(2) = 2 + 2d_0 + c_1 + d_1 = 2$$

and

$$0 = c_1 - d_1$$

Recall $c_1 = 3d_0$ from condition (c). Then

$$0 = 3d_0 - d_1 = -4d_1$$

and

$$d_1 = 0$$

This gives:

$$\begin{aligned} c_0 &= 0 \\ c_1 &= 0 \\ d_0 &= 0 \\ d_1 &= 0 \end{aligned}$$

so far. By condition (a),

$$\begin{aligned} 1 &= b_0 0 \\ 2 &= 1 + b_1 \\ 1 &= b_1 \end{aligned}$$

Then

$$\begin{aligned} S_0(x) &= x \\ S_1(x) &= 1 + (x - 1) = x \end{aligned}$$

Exercise 2: Determine the clamped cubic spline s that interpolates the data $f(0) = 0$, $f(1) = 1$ and $f(2) = 2$ and satisfies $s'(0) = s'(2) = 1$.

Answer. A cubic spline satisfies:

(a) $S_i(x_{i+1}) = S_{i+1}(x_{i+1})$

(b) $S'_i(x_{i+1}) = S'_{i+1}(x_{i+1})$

(c) $S''_i(x_{i+1}) = S''_{i+1}(x_{i+1})$

Since it is clamped, there are additional boundary conditions on the first derivative. Firstly:

$$\begin{aligned} S_0(x) &= b_0 x + c_0 x^2 + d_0 x^3 \\ S_1(x) &= 1 + b_1(x - 1) + c_1(x - 1)^2 + d_1(x - 1)^3 \end{aligned}$$

We have by (a):

$$\begin{aligned} S_0(1) &= b_0 + c_0 + d_0 = 1 \\ S_1(2) &= 1 + b_1 + c_1 + d_1 = 2 \end{aligned}$$

By (b):

$$\begin{aligned} S'_0(1) &= b_0 + 2c_0 + 3d_0 \\ S'_1(1) &= b_1 \end{aligned}$$

are equal. By (c):

$$\begin{aligned} S''_0(1) &= 2c_0 + 6d_0 \\ S''_1(1) &= 2c_1 \end{aligned}$$

are equal. Finally, by the clamped boundary condition:

$$\begin{aligned} S'_0(0) &= b_0 = 1 \\ S'_1(2) &= b_1 + 2c_1 + 3d_1 = 1 \end{aligned}$$

Because $b_0 = 1$, by (a), we have $c_0 + d_0 = 0$. Then by (b), $b_0 + 2c_0 + 3d_0 = b_1$ means that $1 + d_0 = b_1$. By (c), we also have the relation $2c_0 + 6d_0 = 2c_1$ and therefore, $c_1 = 2d_0$. Now by (a), $1 + b_1 + c_1 + d_1 = 2$ means that $d_1 = 1 - b_1 - c_1$, and substituting into $S_1'(2)$,

$$b_1 + 2c_1 + 3(1 - b_1 - c_1) = 1$$

or

$$2 = 2b_1 + c_1$$

Using $b_1 = 1 + d_0, c_1 = 2d_0$,

$$2 = 2(1 + d_0) + 2d_0$$

so

$$2 = 4d_0, d_0 = 0$$

This also gives us

$$\begin{aligned} b_1 &= 1 \\ c_1 &= 0 \end{aligned}$$

We have

$$\begin{aligned} d_1 &= 1 - b_1 - c_1 \\ d_1 &= 0 \end{aligned}$$

Putting this all together:

$$\begin{aligned} S_0(x) &= x \\ S_1(x) &= x \end{aligned}$$