Problem 1.

(25 points)

(a) Use appropriate Lagrange interpolating polynomials of degrees one, two, and three to approximate $f(0.43)$ if

$$f(0) = 1, f(0.25) = 1.64872, f(0.5) = 2.71828, f(0.75) = 4.48169$$

(b) Use Neville's method to obtain the approximations for (a). (You may use Matlab or other softwares to solve this part. Copies of codes and results must be submitted)

(c) The data in (a) is generated using $f(x) = e^{2x}$. Use the error formula to find a bound for the error, and compare the bound to the actual error for the cases n=1 and n=2.

(d) Repeat (a) using Newton divided-difference formula. (You may use Matlab or other softwares to solve this part. Copies of codes and results must be submitted.)

Solution

Part (a)

For the first degree polynomial we use the points 0.25 and 0.5 as we would like to approximate 0.43 within their bounds.

$$\begin{aligned} L_1(x) &= 1.64872\frac{(x-0.5)}{0.25-0.5} + 2.71828\frac{(x-0.25)}{0.5-0.25} \\ &= 4.27824x + 0.57916 \end{aligned}$$

$L_1(0.43) = 2.4188032$

For the second degree polynomial I could either the first or last point, I found that adding the last point produced a more accurate solution, so I'll use that here.

$$\begin{aligned} L_2(x) = &\ 1 \times \frac{(x-0.5)(x-0.75)}{(0.25-0.5)(0.25-0.75)} + 1.64872 \times \frac{(x-0.25)(x-0.75)}{(0.5-0.25)(0.5-0.75)} \\ &+ 2.71828 \times \frac{(x-0.25)(x-0.5)}{(0.75-0.25)(0.75-0.5)} \\ = &\ 5.5508^2 + 0.11514x + 1.27301 \end{aligned}$$

$L_2(0.43) = 2.34886312$

Then for the third degree polynomial, simply use all of the points available.

$$\begin{aligned} L_3(x) = &\ \frac{(x-0.25)(x-0.5)(x-0.75)}{(-0.25)(-0.5)(-0.75)} + 1.64872\frac{(x-0)(x-0.5)(x-0.75)}{(0.25)(0.25-0.5)(0.25-0.75)} \\ &+ 2.71828\frac{(x-0)(x-0.25)(x-0.75)}{(0.5)(0.5-0.25)(0.5-0.75)} + 0.75\frac{(x-0)(x-0.25)(x-0.5)}{(0.75)(0.75-0.25)(0.75-0.5)} \\ = &\ 2.91211x^3 + 1.18264x^2 + 2.11721x + 1 \end{aligned}$$

$L_3(0.43) = 2.36060356577$

Part (b)

$N_1(0.43) =$ 2.41880, $N_2(0.43) =$ 2.34886, $N_3(0.43) =$ 2.36060
Here we can see that the results by Neville's method are equivalent to those produced by Lagrange's method, although the Matlab results produced a slightly lesser degree of accuracy. The code to produce these results can be found at the bottom of this problem.

Part (c)

Firsts, we compute the actual error for each degree of our polynomial. Where $e^{2(0.43)} = 2.36316$ is the actual result.

$$\begin{aligned} \text{degree one: } & |\ 2.4188032 - 2.36316| = 0.0556425 \\ \text{degree two: } & |\ 2.34886312 - 2.36316| = 0.01429688 \\ \text{degree three: } & |\ 2.36060356577 - 2.36316| = 0.00255643423 \end{aligned}$$

The Lagrange polynomial error term is given by

$$|\frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n)|$$

To find maximum error, we would like to maximize the term: $|f^{(n+1)}(\xi(x))|$.

For degree one we want the maximum on the bounds $(0.25, 0.5)$:
$\max(|f''(\xi(x))|) = \max(|4e^{2(\xi(x))}|) = |4e^{2*0.5}| = 10.8731$.
With just a little more work we have the maximum error as
$|\frac{10.8731}{2!}(0.43-0.25)(0.43-0.5)|=$ 0.0685. We can see that our actual error of 0.0556425 is just squeezing underneath this maximal value.

Next for degree two, we want the maximum on the bounds $(0.25, 0.75)$:
$\max(|f^{(3)}(\xi(x))|) = \max(|8e^{2(\xi(x))}|) = |8e^{2*0.75}| = 35.8535$.
Then to compute maximum error, $|\frac{33.8535}{3!}(0.43-0.25)(0.43-0.5)(0.43-0.75)|=$ 0.0225877. This error is obviously improved accuracy over degree one, and again we see that our actual error of 0.01429688 is well within its bounds.

Finally for degree three, we want the maximum on the full bounds $(0, 0.75)$:
$\max(|f^{(4)}(\xi(x))|) = \max(|16e^{2(\xi(x))}|) = |16e^{2*0.75}| = 71.7070$.
Then to compute maximum error, $|\frac{71.7070}{4!}(0.43-0)(0.43-0.25)(0.43-0.5)(0.43-0.75)|=$ 0.00485636. Again, this behaves as we'd expect, less error than degree two, and our actual error of 0.00255643423 is again well within its bounds.

Part (d)

$\text{ndd}_1(0.43) = 2.4188$
$\text{ndd}_2(0.43) = 2.3489$
$\text{ndd}_3(0.43) = 2.3764$

```
function p2

x = 0.43;
xi = [0 0.25 0.5 0.75];
fi = [1 1.64872 2.71828 4.48169];

%% Neville's Method

% degree one
neville(x, xi(:,2:3), fi(:,2:3))
% degree two
neville(x, xi(:,2:4), fi(:,2:4))
% degree three
neville(x, xi(:,1:4), fi(:,1:4))

%% Newton's Divided-Difference

% degree one
ndd(x, xi(:,2:3), fi(:,2:3))
% degree two
ndd(x, xi(:,2:4), fi(:,2:4))
% degree three
ndd(x, xi(:,1:3), fi(:,1:3))

end
```

```
function res = ndd(x, xi, fi)

M = divideddifference(xi, fi);
n = length(M);
res = 0;

X = 1;
for i=1:n
        C =  M(i,i);
        res = res + C * X;
        X = X * (x - xi(i));
end

end
```

```
function F = divideddifference(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
```

```
function Q = neville(x, xi, fi)

n = length(xi) - 1;
Q = zeros(n+1,n+1);
Q(:,1) = fi(:);

for i=1:n
    for j=1:i
        Q(i+1,j+1) = ((x-xi(i-j+1))*Q(i+1,j)-(x-xi(i+1))*Q(i,j))...
            /(xi(i+1)-xi(i-j+1));
    end
end
```

Problem 2.

(25 points)
The Bernstein polynomial of degree n for $f \in C[0,1]$ is given by

$$B_n(x) = \Sigma_{k=0}^{n} \binom{n}{k} f(k/n) x^k (1-x)^{n-k},$$

where $\binom{n}{k} = n!\,/(k!\,(n-k)!\,)$. These polynomials can be used in a constructive proof of Weierstrass Approximation Theorem since $\lim_{n\to\infty} B_n(x) = f(x)$, for each $x \in [0,1]$.

(a) Find $B_3(x)$ for the functions: (i) $f(x) = x$ and (ii) $f(x) = 1$.

(b) Show that for each $k \leq n$,

$$\binom{n-1}{k-1} = \frac{k}{n}\binom{n}{k}$$

(c) Use part (b) and the fact, from (ii) in part (a), that

$$1 = \Sigma_{k=0}^{n} \binom{n}{k} x^k (1-x)^{n-k}, \forall n \in \mathbb{N},$$

to show that for $f(x) = x^2$,

$$B_n(x) = \frac{n-1}{n} x^2 + \frac{1}{n} x.$$

(d) Use part (c) to estimate the value of n necessary for $|B_n(x) - x^2| \leq 10^{-6}$ to hold for all $x \in [0,1]$.

Solution

Part (a)

f(x) = x

$$\begin{aligned}
B_3(x) &= \binom{3}{0} f(0/3) x^0 (1-x)^{3-0} + \binom{3}{1} f(1/3) x^1 (1-x)^{3-1} \\
&+ \binom{3}{2} f(2/3) x^2 (1-x)^{3-2} + \binom{3}{3} f(3/3) x^3 (1-x)^{3-3} \\
&= 0 \times x^0 (1-x)^3 + 3 \times 1/3 x^1 (1-x)^2 + 3 \times 2/3 x^2 (1-x)^1 + x^3 (1-x)^0 \\
&= x(1-x)^2 + 2x^2(1-x) + x^3 \\
&= x - 2x^2 + x^3 + 2x^2 - 2x^3 + x^3 \\
&= x - 2x^2 + 2x^2 + 2x^3 - 2x^3 \\
&= x
\end{aligned}$$

f(x) = 1

$$\begin{aligned}
B_3(x) &= \binom{3}{0} f(0/3)x^0(1-x)^{3-0} + \binom{3}{1} f(1/3)x^1(1-x)^{3-1} \\
&+ \binom{3}{2} f(2/3)x^2(1-x)^{3-2} + \binom{3}{3} f(3/3)x^3(1-x)^{3-3} \\
&= x^0(1-x)^3 + 3x^1(1-x)^2 + 3x^2(1-x)^1 + x^3(1-x)^0 \\
&= -x^3 + 3x^2 - 3x + 1 + 3x(1 - 2x + x^2) + 3x^2(1-x) + x^3 \\
&= -x^3 + 3x^2 - 3x + 1 + 3x - 6x^2 + 3x^3 + 3x^2 - 3x^3 + x^3 \\
&= (-x^3 + x^3 + 3x^3 - 3x^3) + (3x^2 - 6x^2 + 3x^2) + (-3x + 3x) + 1 \\
&= 1
\end{aligned}$$

Part (b)

Proof. Let $k \leq n$.

$$\begin{aligned}
\binom{n-1}{k-1} &= \frac{(n-1)!}{(k-1)!\,((n-1)-(k-1))!} = \frac{(n-1)!}{(k-1)!\,(n-k)!} \\
&= \frac{k}{k}\frac{(n-1)!}{(k-1)!\,(n-k)!} = \frac{k(n-1)!}{k!\,(n-k)!} \\
&= \frac{n}{n}\frac{k(n-1)!}{k!\,(n-k)!} = \frac{kn(n-1)!}{nk!\,(n-k)!} = \frac{kn!}{nk!\,(n-k)!} \\
&= \frac{k}{n}\frac{n!}{k!\,(n-k)!} = \frac{k}{n}\binom{n}{k}
\end{aligned}$$

□

Part (c)

Proof.

$$\begin{aligned}
B_n(x) &= \Sigma_{k=0}^{n}\binom{n}{k}f(k/n)x^k(1-x)^{n-k} \\
&= \Sigma_{k=0}^{n}\binom{n}{k}\frac{k^2}{n^2}x^k(1-x)^{n-k} \\
&= (\Sigma_{k=0}^{1}\binom{n}{k}\frac{k^2}{n^2}x^k(1-x)^{n-k}) + (\Sigma_{k=2}^{n}\binom{n-2}{k-2}x^k(1-x)^{n-k}) \\
&= (\Sigma_{k=0}^{1}\binom{n}{k}\frac{k^2}{n^2}x^k(1-x)^{n-k}) + x^2(\Sigma_{k=2}^{n}\binom{n-2}{k-2}x^{k-2}(1-x)^{(n-2)-(k-2)}) \\
&= (\Sigma_{k=0}^{1}\binom{n}{k}\frac{k^2}{n^2}x^k(1-x)^{n-k}) + x^2(\Sigma_{k=0}^{n-2}\binom{n}{k}x^k(1-x)^{n-k}) \\
&= (\Sigma_{k=0}^{1}\binom{n}{k}\frac{k^2}{n^2}x^k(1-x)^{n-k}) + x^2 \\
&= \binom{n}{1}\frac{1^2}{n^2}x(1-x)^{n-1} + x^2 \\
&= \frac{n}{n^2}x(1-x)^{n-1} + x^2 \\
&= \frac{x}{n}(1-x)^{n-1} + x^2
\end{aligned}$$

□

Part (d)

$$\begin{aligned}
|\frac{n-1}{n}x^2 + \frac{1}{n}x - x^2| \leq 10^{-6} \Rightarrow \\
|(\frac{n-1}{n} - \frac{n}{n})x^2 + \frac{x}{n}| \leq 10^{-6} \Rightarrow \\
|\frac{-x^2}{n} + \frac{x}{n}| \leq 10^{-6} \Rightarrow \\
|\frac{x-x^2}{n}| \leq 10^{-6} \Rightarrow \\
\frac{1}{n}|x-x^2| \leq 10^{-6}
\end{aligned}$$

We want this to hold $\forall x \in [0,1]$ thus we need the maximum value of $|x-x^2|$ on those bounds. Thus is found at $x = 0.5$ with $|0.5 - 0.5^2| = 0.25$. Thus we want $\frac{1}{4n} \leq 10^{-6} \Rightarrow 4n \geq 10^6 \Rightarrow n \geq \frac{10^6}{4} \Rightarrow n \geq$ 250000.

Problem 3.

(10 points)

(a) Show that the cubic polynomials

$$P(x) = 3 - 2(x+1) + 0(x+1)(x) + (x+1)(x)(x-1)$$

and

$$Q(x) = -1 + 4(x+2) - 3(x+2)(x+1) + (x+2)(x+1)(x)$$

both interpolate the data $f(-2) = -1, f(-1) = 3, f(0) = 1, f(1) = -1, f(2) = 3$

(b) Why does part (a) not violate the uniqueness property of interpolating polynomials?

Solution

Part (a)

Using the MATLAB code provided below, we can see that both $P(x)$ and $Q(x)$ produce the expected results. That is $P(-2) = Q(-2) = f(-2) = -1, P(-1) = Q(-1) = f(-1) = 3, P(0) = Q(0) = f(0) = 1, P(1) = Q(1) = f(1) = -1$, and $P(2) = Q(2) = f(2) = 3$. Thus the given cubic polynomials both clearly interpolate the provided points of f.

```
function p3

P = @(x) 3 - 2.*(x+1) + 0.*(x+1).*(x) + (x+1) .* x .* (x-1);
Q = @(x) -1 + 4 .* (x+2) - 3 .* (x+2).*(x+1) + (x+2).*(x+1).*(x);
X = -2:2;
P(X)
Q(X)

end
```

Part (b)

Because 5 interpolating points are provided for f, where as P and Q only have degree 3. The uniqueness property of interpolating polynomials would only holds for polynomials of degree 4.

Problem 4.

(20 points)
(You may use Matlab or other softwares to solve this problem. Copies of codes and results must be submitted.) Let $f(x) = 3xe^x - e^{2x}$.

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

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

Solution

Part (a)

The actual result is $f(1.03) = 0.80932$ with the Hermite estimate being $H(1.03) = 0.80932$. At the precision MATLAB outputs these seem nearly identical, however the code which computes the error gives us a small error between the two results with ERROR $= 1.2373 \times 10^{-6}$.
The error term of the Hermite polynomial is given by $|\frac{f^{(2n+2)\xi(x)}}{(2n+2)!}\Pi_{k=0}^{n}(x-x_i)^2|$ We only have $n = 1$, so we will need the 4th derivative, this is given by $f^{(4)}(x) = e^x(3(x+4) - 16e^x)$. Maximizing error requires maximizing the magnitude on our bounds $[1, 1.05]$. This is found at $x = 1.05$ where $|f^(4)(1.05)| = 87.3653$. Thus, our maximum error will be

$$|\frac{87.3653}{4!}(1.03-1)^2(1.03-1.05)^2| = 1.3104795 \times 10^{-6}$$

This is very close to the actual error we discovered, about 1.2×10^{-6} actual versus the 1.3×10^{-6} maximum theoretical.

```
function p4a

f = @(t) 3.*t.*exp(t) - exp(2.*t);
df= @(t) exp(t) .* (3.*t - 2.*exp(t) + 3);

x = [1 1.05];
y = f(x);
d = df(x);

Q = Hermiteinterpolation(x,y,d);

H = @(t) Q(1,1) + Q(2,2)*(t-x(1)) + ...
        Q(3,3)*(t-x(1))^2 + ...
        Q(4,4)*(t-x(1))^2*(t-x(2));

% Compute our results
actual = f(1.03)
```

```
est = H(1.03)
ERR = abs(actual - est)

end
```

Part (b)

Again the actual result of $f(1.03) = 0.80932$ is equal to the Hermite estimate of $H(1.03) = 0.80932$ due to the limitations of MATLAB output. But when we produce an error, we see that it has improved over the result in part (a), with an error of only 3.6101×10^{-10}.
The error term of the Hermite polynomial is given by $|\frac{f^{(2n+2)\xi(x)}}{(2n+2)!}\Pi_{k=0}^{n}(x-x_i)^2|$ This time we have $n = 2$, so we will need the 6th derivative, this is given by $f^{(6)}(x) = 6e^x(x+6) - 64e^{2x}$. Maximizing error requires maximizing the magnitude on our bounds $[1, 1.07]$. This is found at $x = 1.07$ where $|f^(6)(1.07)| = 420.294$. Thus, our maximum error will be

$$|\frac{420.294}{6!}(1.03-1)^2(1.03-1.05)^2(1.03-1.07)^2| = 1.73699 \times 10^{-4}$$

Although the actual error did decrease from part (a) to part (b), widening the bounds did actually increase the maximum theoretical error.

```
function p4b

f = @(t) 3.*t.*exp(t) - exp(2.*t);
df= @(t) exp(t) .* (3.*t - 2.*exp(t) + 3);

x = [1 1.05 1.07];
y = f(x);
d = df(x);

Q = Hermiteinterpolation(x,y,d)

H = @(t) Q(1,1) + Q(2,2)*(t-x(1)) + ...
        Q(3,3)*(t-x(1))^2 + ...
        Q(4,4)*(t-x(1))^2*(t-x(2)) + ...
        Q(5,5)*(t-x(1))^2*(t-x(2))^2 + ...
        Q(6,6)*(t-x(1))^2*(t-x(2))^2*(t-x(3));

% Compute our results
actual = f(1.03)
est = H(1.03)
ERR = abs(actual - est)

end
```

```
function Q = Hermiteinterpolation(x,f,df)

x = x(:); f = f(:); df = df(:);

n = length(x);
Q = zeros(2*n,2*n);

z = zeros(2*n,1);
z(1:2:end-1) = x;
z(2:2:end) = x;

Q(1:2:end-1,1) = f;
Q(2:2:end,1) = f;
Q(2:2:end,2) = df;

Q(3:2:end-1,2) = (Q(3:2:end-1,1) - Q(2:2:end-2,1))./(z(3:2:end-1)-z(2:2:end-2));


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
```

Problem 5.

(20 points)

(a) Determine the free cubic spline S that interpolates the data $f(0) = 0$, $f(1) = 1$ and $f(2) = 2$.

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

Solution

Part (a)

Using the MATLAB code included below, we can produce the coefficients for each of the cubic polynomials which define the natural cubic spline S. S is then given by

$$S(x) = \begin{cases} 0 + 1x + 0x^2 + 0x^3 : 0 \leq x \leq 1 \\ 1 + 1x + 0x^2 + 0x^3 : 1 < x \leq 2 \end{cases}$$

Part (b)

Running similar code, but for the clamped cubic spline with derivatives of 1 produces the same piece-wise cubic polynomials.

$$S(x) = \begin{cases} 0 + 1x + 0x^2 + 0x^3 : 0 \leq x \leq 1 \\ 1 + 1x + 0x^2 + 0x^3 : 1 < x \leq 2 \end{cases}$$

```
x = [0 1 2];
y = [0 1 2];
v = [1 1];

%% Natural Cubic Spline
p_nat = csape(x,y,'variational');
p_nat.coefs % Display the coefficients

%% Clamped Cubic Spline
p_cla = csape(x, y, 'clamped', v);
p_cla.coefs % Display the coefficients
```