1. (35 points) The following MATLAB code generates the free/natural cubic spline coefficients given x and f data:

```
function M=CubicSpline19(x,f)
n=length(x)-1;
h=zeros(n,1);
h=x(2:n+1,1)-x(1:n,1);
C=zeros(n+1);
C(1,1)=1;
C(n+1,n+1)=1;
for j=2:n
    C(j,j-1)=h(j-1);
    C(j,j)=2*(h(j-1)+h(j));
    C(j,j+1)=h(j);
end
v=zeros(n+1,1);
for j=2:n
    v(j)=3/h(j)*(f(j+1)-f(j))-3/h(j-1)*(f(j)-f(j-1));
end
c=C^-1*v;
a=f;
d=zeros(n+1,1);
for i=1:n
    d(i)=(c(i+1)-c(i))/(3*h(i));
end
b=zeros(n+1,1);
for i=1:n
    b(i)=(f(i+1)-f(i))/(h(i))-h(i)*(c(i+1)+2*c(i))/3;
end
M=[a,b,c,d];
```

Adjust this code to provide the coefficients for the clamped cubic spline. You may assume that $f'(x_0)$ and $f'(x_n)$ would be additional data given by the user.

2. (35 points) The following MATLAB code derives the 5 point midpoint formula:

```
function FivePointMidPoint()

syms c h f0 f1 f2 f3 f4 x
% c denotes the midpoint
% h denotes the spacing between consecutive values of x
% f0, f1, f2, f3, and f4 are the function values at x0, x1, x2, x3, ...
    and x4
% (respectively).

% the following vector contains all of the x values
v=c*[1;1;1;1;1]+h*[-2;-1;0;1;2];

% the following vector contains all of the y values
f=[f0;f1;f2;f3;f4];
```

```
14
15 % we will build the Lagrange coefficient polynomials
16 LCP=[];
17 for n=0:4
18     L=1;
19     for j=0:4
20         if j≠n
21             L=L*(x-v(j+1))/(v(n+1)-v(j+1));
22         end
23     end
24     LCP=[LCP;L];
25 end
26
27 % we now express the Lagrange Interpolating Polynomial as an ...
      appropriate linear combination of the coefficient polynomials
28 L=sum(LCP.*f);
29
30 % we differentiate the Lagrange Interpolating Polynomial
31 dL=diff(L,x);
32
33 % the formula comes from evaluating this derivative at c
34 formula=subs(dL,c)
```

Adjust this code to provide seven point midpoint and endpoint formulae.

Solution:

The following code will provide the seven point mid point formula.

```
1  function SevenPointMidPoint()
2
3  syms c h f0 f1 f2 f3 f4 f5 f6 x
4  % c denotes the midpoint
5  % h denotes the spacing between consecutive values of x
6  % f0, f1, f2, f3, f4, f5 and f6 are the function values at x0, ...
      x1, x2, x3 x4, x5 and x6
7  % (respectively).
8
9  % the following vector contains all of the x values
10 v=c*[1;1;1;1;1;1;1]+h*[-3;-2;-1;0;1;2;3];
11
12 % the following vector contains all of the y values
13 f=[f0;f1;f2;f3;f4;f5;f6];
14
15 % we will build the Lagrange coefficient polynomials
16 LCP=[];
17 for n=0:6
18     L=1;
19     for j=0:6
20         if j≠n
```

```
            L=L*(x-v(j+1))/(v(n+1)-v(j+1));
        end
    end
    LCP=[LCP;L];
end

% we now express the Lagrange Interpolating Polynomial as an ...
   appropriate linear combination of the coefficient polynomials
L=sum(LCP.*f);
% we differentiate the Lagrange Interpolating Polynomial
dL=diff(L,x);
% the formula comes from evaluating this derivative at c
formula=subs(dL,c)
```

This gives the formula:

$$\frac{(3_1)}{(20h)} - \frac{f_0}{(60h)} - \frac{(3f_2)}{(4h)} + \frac{(3f_4)}{(4h)} - \frac{(3f_5)}{(20h)} + \frac{f_6}{(60h)}$$

The following code will provide the seven point end point formula.

```
function SevenPointMidPoint()

syms c h f0 f1 f2 f3 f4 f5 f6 x
% c denotes the endpoint
% h denotes the spacing between consecutive values of x
% f0, f1, f2, f3, f4, f5 and f6 are the function values at x0, ...
   x1, x2, x3 x4, x5 and x6
% (respectively).

% the following vector contains all of the x values
v=c*[1;1;1;1;1;1;1]+h*[0;1;2;3;4;5;6];

% the following vector contains all of the y values
f=[f0;f1;f2;f3;f4;f5;f6];

% we will build the Lagrange coefficient polynomials
LCP=[];
for n=0:6
    L=1;
    for j=0:6
        if j ≠ n
            L=L*(x-v(j+1))/(v(n+1)-v(j+1));
        end
    end
    LCP=[LCP;L];
end

% we now express the Lagrange Interpolating Polynomial as an ...
   appropriate linear combination of the coefficient polynomials
```

```
L=sum(LCP.*f);
% we differentiate the Lagrange Interpolating Polynomial
dL=diff(L,x);
% the formula comes from evaluating this derivative at c
formula=subs(dL,c)
```

This gives the formula:

$$\frac{(6f_1)}{h} - \frac{(49f_0)}{(20h)} - \frac{(15f_2)}{(2h)} + \frac{(20f_3)}{(3h)} - \frac{(15f_4)}{(4h)} + \frac{(6f_5)}{(5h)} - \frac{f_6}{(6h)}$$

3. (30 points) Use the 7 point midpoint formula derived in the previous question to estimate $f'(2.4)$, where $f(x) = x\sin(x)$, and the function is sampled at

$$x = 2.1,\ 2.2,\ 2.3,\ 2.4,\ 2.5,\ 2.6,\ 2.7.$$

Determine the absolute error in this approximation. You may use MATLAB here.

Solution: We used the following code to calcuate the error.

```
syms f(x);
f(x) = x * sin(x);
fp(x) = diff(f, x);

spm = (3*f(2.2))/(20*.1) - f(2.1)/(60*.1) - (3*f(2.3))/(4*.1) + ...
    (3*f(2.5))/(4*.1) - (3*f(2.6))/(20*.1) + f(2.7)/(60*.1);
vpa(spm)
vpa(fp(2.4))

err = abs(vpa(spm) - vpa(fp(2.4)))
```

This shows the absolute error in using the seven point midpoint formula to approximate $2.4 * sin(2.4)$ is

$$0.00000002107234578242043477854908439584 7$$