Nathaniel Valla

```
format long
clc
clear
```

1. Given the data points (1,0),(2,ln 2),(4,ln 4)

(a) Determine the degree 2 interpolating polynomial.

```
syms x
x_i= [1,2,4];
y_i=[log(1),log(2),log(4)];
xf = 3;
equ = newtons_div_dff(x_i,y_i)
```

equ = $0.6931\,x - 0.1155\,(x-1)\,(x-2) - 0.6931$

(b) Use the result of (a) to approximate ln 3.

```
approx = subs(equ,x,3)
```

approx = 1.1552

(c) Give an error bound for the approximation in part (b).

```
err_equ = poly_error(x_i,log(x))
```

err_equ =

$$\frac{0.3333\,(x-1)\,(x-2)\,(x-4)}{x^3}$$

```
subs(err_equ,x,3)
```

ans = -0.0247

(d) Compare the actual error (assuming the ln function) to your error bound.

```
real = log(3);
abs(real-approx)
```

ans = 0.0566

2. In addition to using Lagrange polynomials to estimate values between points, they could also be used to estimate derivatives.

(a) Consider a function, f , and three points x0, x1, and x2. Estimate the derivative of f at x0 by differentialing the Lagrange polynomial through the three points. That is, find p′ 3 (x0) along with the error estimate.

```
syms x
x_i = sym('x', [1 3]);
```

```
y_i = sym('y', [1 3]);
equ = newtons_div_dff(x_i,y_i)
```

equ =

$$y_1 + \frac{(x - x_1)\ (y_1 - y_2)}{x_1 - x_2} + \frac{\left(\frac{y_1 - y_2}{x_1 - x_2} - \frac{y_2 - y_3}{x_2 - x_3}\right)\ (x - x_1)\ (x - x_2)}{x_1 - x_3}$$

```
equ_p = diff(equ,x)
```

equ_p =

$$\frac{y_1 - y_2}{x_1 - x_2} + \frac{\left(\frac{y_1 - y_2}{x_1 - x_2} - \frac{y_2 - y_3}{x_2 - x_3}\right)\ (x - x_1)}{x_1 - x_3} + \frac{\left(\frac{y_1 - y_2}{x_1 - x_2} - \frac{y_2 - y_3}{x_2 - x_3}\right)\ (x - x_2)}{x_1 - x_3}$$

```
poly_error(x_i,"null")
```

ans = $0.1667\ \mathrm{df}_n\ (x - x_1)\ (x - x_2)\ (x - x_3)$

```
equ_p_1= subs(equ_p,x,x_i(1))
```

equ_p_1 =

$$\frac{y_1 - y_2}{x_1 - x_2} + \frac{\left(\frac{y_1 - y_2}{x_1 - x_2} - \frac{y_2 - y_3}{x_2 - x_3}\right)\ (x_1 - x_2)}{x_1 - x_3}$$

```
poly_error(x_i,"one")
```

ans = $\mathrm{df}_n\ (0.1667\ (x - x_1)\ (x - x_2) + 0.1667\ (x - x_1)\ (x - x_3) + 0.1667\ (x - x_2)\ (x - x_3)) + 0.1667\ \mathrm{df}_{n1}\ (x - x_1)$

(b) Now assume that xi = x+ih, for i = 0,1,2. Plug this in to your result from (a) to derive a three-point formula for the first derivative (along with the error). How does the error here compare to the error you see from the typical two-point estimate of the derivative you learned in Calc I? in particular, discuss how accuracy is affected in each case by changes in h.

```
h_i = sym('h', [1 3]);
x_j = (x-h_i);
subs(equ_p_1,x_i,x_j)
```

ans =

$$-\frac{y_1 - y_2}{h_1 - h_2} - \frac{\left(\frac{y_1 - y_2}{h_1 - h_2} - \frac{y_2 - y_3}{h_2 - h_3}\right)\ (h_1 - h_2)}{h_1 - h_3}$$

```
poly_error(x_j,"one")
```

ans = $0.1667\ \mathrm{df}_{n1}\ h_1\ h_2\ h_3$

```
poly_error(x_j,"null")
```

```
ans = 0.1667 df_n h1 h2 h3
```

(c) In general, asubs(equ_p,x,x_i(1))e the derivatives from the Lagrange polynomials going to be more or less accurate than the function values? Give some explanation for your answer, though formal proof is not required

The error is the same for the direvative and normal function expect fo the for the error your multipling the n plus one derivate. so as long as the n derivate is large then n+1 derivate will be better.

3) Determine k1, k2, k3 in the following cubic spline. Which of the three end conditions—natural, parabolically terminated, or not-a-knot—if any, are satisfied?

$$X(m,n) = \begin{cases} 4 + k_1 x + 2x^2 - \frac{1}{6}x^3 & on[0.1] \\ 1 - \frac{4}{3}(x-1) + k_2(x-1)^2 - \frac{1}{6}(x-1)^3 & on[1,2] \\ 1 - k_3(x-2) + (x-2)^2 - \frac{1}{6}(x-2)^3 & on[2,3] \end{cases}$$

$$f_0(x) = 4 + k_1 x + 2x^2 - \frac{1}{6}x^3$$

$$f_1(x) = 1 - \frac{4}{3}(x-1) + k_2(x-1)^2 - \frac{1}{6}(x-1)^3$$

$$f_2(x) = 1 + k_3(x-2) + (x-2)^2 - \frac{1}{6}(x-2)^3$$

$$f_0(1) = f_1(1) \rightarrow 4 + k_1(1) + 2(1)^2 - \frac{1}{6}(1)^3 = 1 \rightarrow k_1 = -4 - 2 + \frac{7}{6} \rightarrow k_1 = -\frac{29}{6}$$

$$f_0(x) = 4 - \frac{29}{6}x + 2x^2 - \frac{1}{6}x^3$$

$$f_1(2) = f_2(2) \rightarrow 1 - \frac{4}{3}(1) + k_2(1)^2 - \frac{1}{6}(1)^3 = 1 \rightarrow k_2 = \frac{4}{3} + \frac{1}{6} \rightarrow k_2 = \frac{3}{2}$$

$$f_1(x) = 1 - \frac{4}{3}(x-1) + \frac{3}{2}(x-1)^2 - \frac{1}{6}(x-1)^3$$

$$f'_1(x) = -\frac{4}{3} + 3(x-1) - \frac{1}{2}(x-1)^2$$

$$f'_2(x) = k_3 + 2(x-2) - \frac{1}{2}(x-2)^2$$

$$f'_1(2) = f'_2(2) \rightarrow -\frac{4}{3} + 3(1) - \frac{1}{2}(1)^2 = k_3 \rightarrow k_3 = -\frac{4}{3} + 3 - \frac{1}{2} \rightarrow k_3 = \frac{7}{6}$$

$$f_2(x) = 1 + \frac{7}{6}(x-2) + (x-2)^2 - \frac{1}{6}(x-2)^3$$

$$k_1 = \frac{-29}{6}, k_2 = \frac{3}{2}, k_3 = \frac{7}{6}$$

It is not a parabolic termiting the ends are paralobes.

$$f_1(x) = 1 - \frac{4}{3}(x-1) + \frac{3}{2}(x-1)^2 - \frac{1}{6}(x-1)^3$$

$$f'_1(x) = -\frac{4}{3} + 3(x-1) - \frac{1}{2}(x-1)^2$$

$$f''_1(x) = 4 - x$$

$$f'''_1(x) = -1$$

$$f''_1(2) = 4 - 2 = 2$$

$$f'_1(2) = \frac{4}{3} + 3(2-1) - \frac{1}{2}(2-1)^2 = \frac{4}{3} + 3 - \frac{1}{2} = \frac{7}{6}$$

Not Natural cause second dervative is not equal to 0

$$f'_2(x) = \frac{7}{6} + 2(x-2) - \frac{1}{2}(x-2)^2$$

$$f''_2(x) = 2 - (x-2)$$

$$f_2''(x) = -2$$

$$f''_2(2) = 2 - (2-2) = 2$$

$$f'_2(2) = \frac{7}{6} + 2(2-2) - \frac{1}{2}(2-2)^2 = \frac{7}{6}$$

Not a Knot caus ehtird Dervivetsare the same

Clamped cuase first derviates are not zero

Curvature second derviates are not zero

```
function sol = newtons_div_dff(x_i,y_i)
    syms x
    n=length(x_i); % Number of terms of X or Y
    D=zeros(n,'sym');% For divided difference functions.
    for k=1:n
        D(k,1)=y_i(k);
    end

    for k=2:n
        for m=1:n-k+1
            D(m,k)=(D(m,k-1)-D(m+1,k-1))/(x_i(m)-x_i(m+(k-1)));
        end
    end

    %disp(L);

    s=1;
    sol=y_i(1);
    for k=2:n
        s=s*(x-x_i(k-1));
        t=s*D(1,k);
        sol=sol+t;
    end
    %fprintf('The required value is f(%1.2f)= %2.3f',xf,sol);
end

function err_bound = poly_error(x_i,fun)
    syms x df_n df_n1
    n=length(x_i); % Number of terms of X or Y

    s=1;
    for k=1:n
        s=s*(x-x_i(k));
    end
    if (fun == "null")
        err_bound = s/factorial(n)*df_n;
    elseif(fun == "one")
        err_bound = s/factorial(n)*df_n1+diff(s,x)/factorial(n)*df_n;
    else
        err_bound = s*diff(fun,x,n)/factorial(n);
    end
    %fprintf('The required value is f(%1.2f)= %2.3f',xf,sol);
end
```