2021

Homework 6

AE_5830 DR. HOSDER

MATT PAHAYO

Use 3^{rd} and 6^{th} degree Lagrange interpolation polynomials to approximate the function

$$f(x) = Sin\,(e^x - 2)$$

using equally spaced data on the interval $0 \leq x \leq 2$. For each case, plot the exact function $f(x)$, approximation to the function by the Lagrange polynomials ($f_{appx}(x)$), and the error ($f(x) - f_{appx}(x)$) on the given interval. Give the error values at x=0.1, 0.9, 1.5, and 1.9.

Results

Q1:

Table 1 Error of at specific values per degree of polynomial

x	3rd	6th
0.1	-0.220766823	-0.139989157
0.9	-0.196722589	-0.034689456
1.5	0.306661865	0.071951521
1.9	-0.495939913	-0.179944255

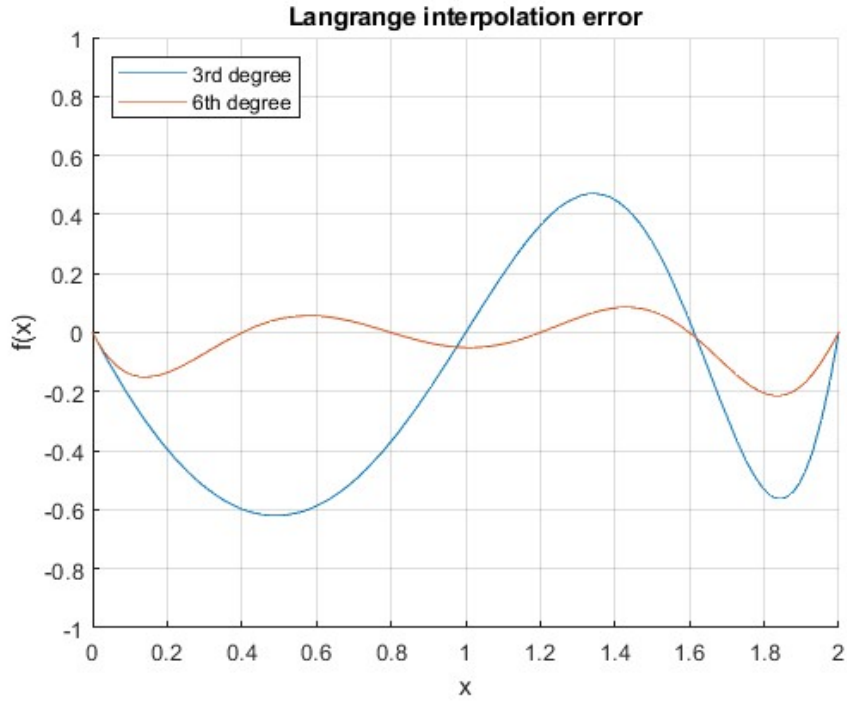


Figure 1 Absolute Error plot

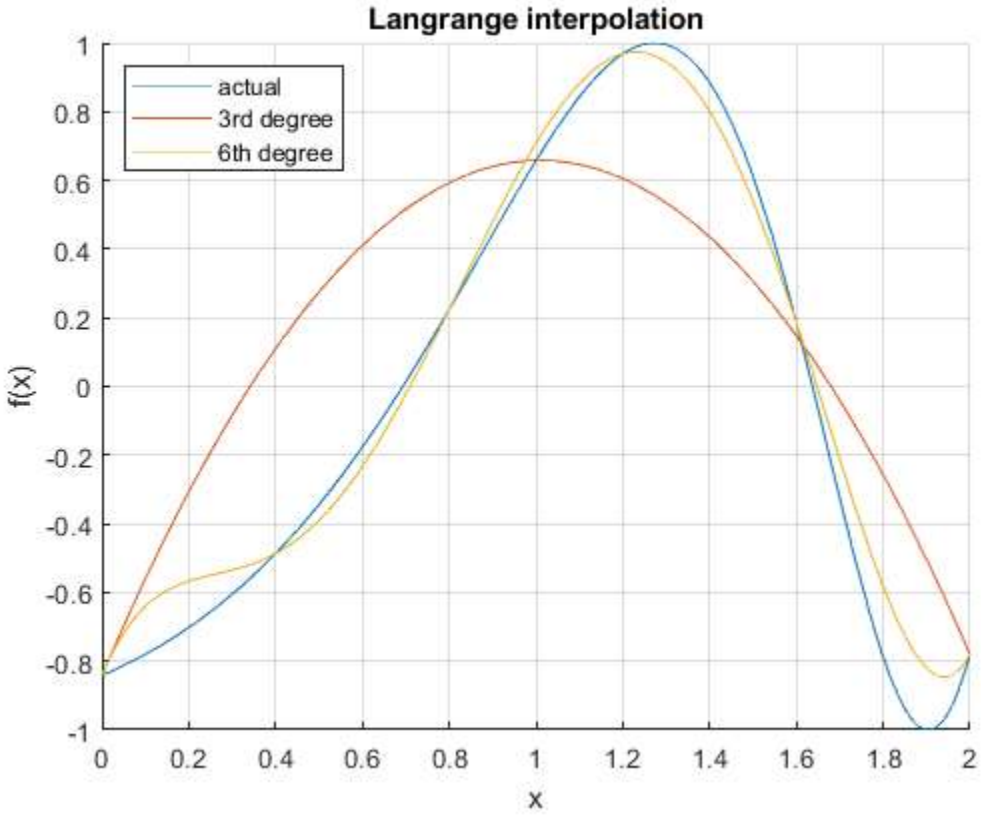


Figure 2 plot of approximations and function

(1) $$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

(2) $$f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$$

(3) n =3 and 6

Q2:

Write a computer routine to approximate the function given in Question 1 using a natural cubic spline with eleven equally spaced data points on the interval $0 \leq x \leq 2$. Plot the function, the cubic spline approximation ($f_{appx}(x)$), and the error ($f(x) - f_{appx}(x)$) on the given interval. Give the error values at x=0.1, 0.9, 1.5, and 1.9.

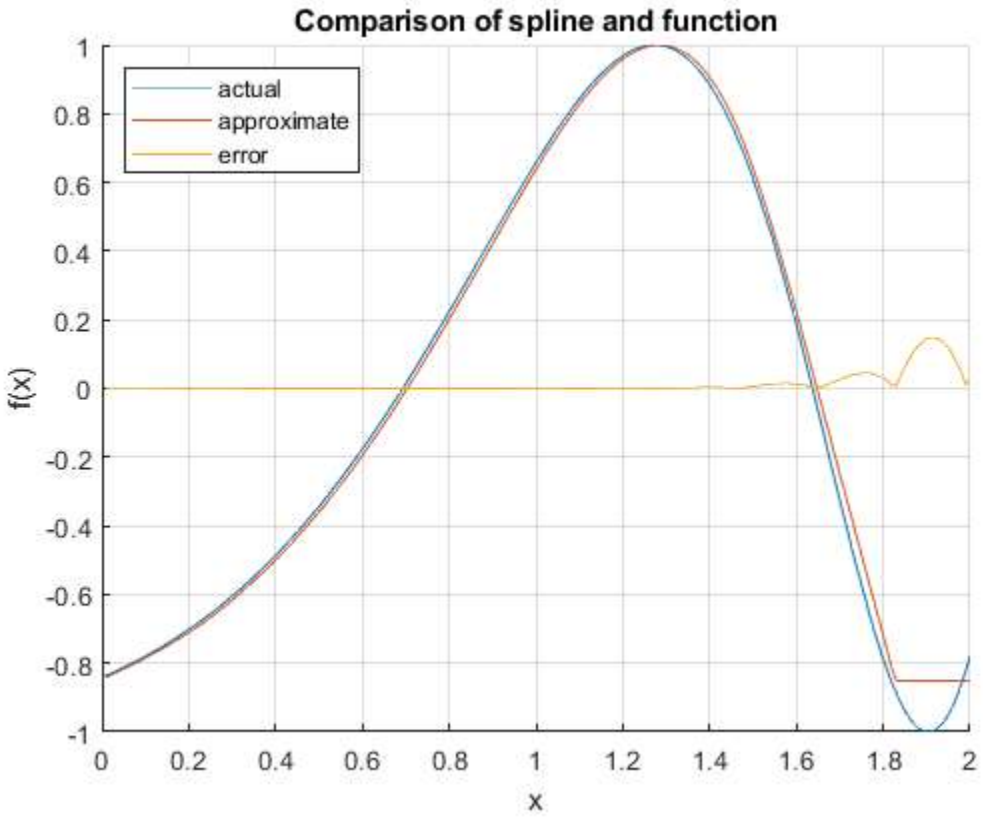


Figure 3 Q2 plot

x	spline error
0.1	0.002085
0.9	1.65E-05
1.5	0.006003
1.9	0.1441

(1) Find coeffcients by solving a linear system for a natural cubic spline
(2) Plug back into cubic spline quation

Q3:

Derive the following central finite difference formula

$$\left(\frac{df}{dx}\right)_i = \frac{1}{12h}\left(-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}\right)$$

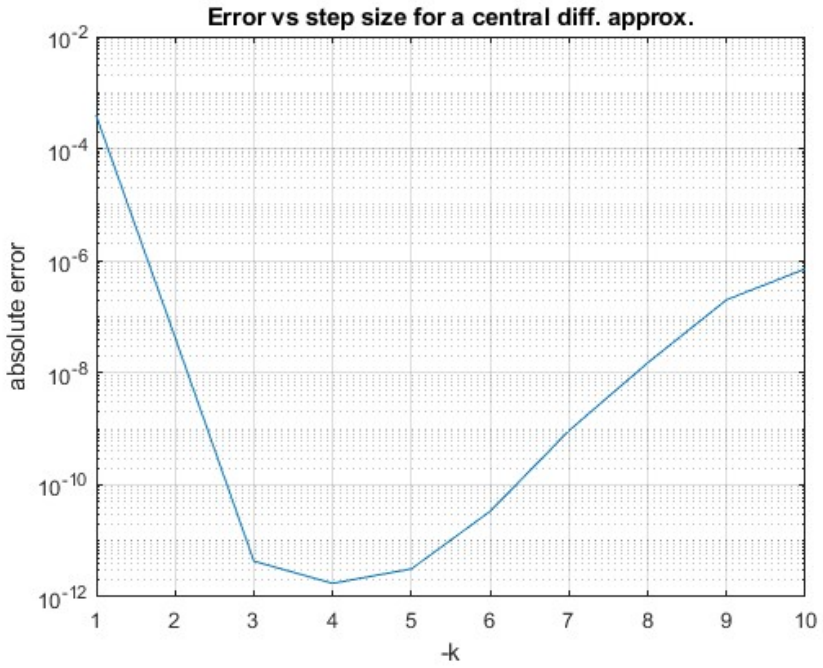
to approximate the first derivative of the function at x_i. Note that $f_{i+2} = f(x_i + 2h)$, $f_{i+1} = f(x_i + h)$, $f_{i-1} = f(x_i - h)$, and $f_{i-2} = f(x_i - 2h)$. Show that the order of this approximation is $O(h^4)$.

(i) Use the above finite difference formula to calculate the first derivative of the function given in Question 1 at $x_i = 0.9$ with $h = 10^{-k}$ for $k = 1, 2, 3....., 10$. Calculate the absolute error for each case using the exact value of the derivative at the given point. Comment on the results you obtain.

(ii) Now calculate the first derivative of the function given in Question 1 at $x_i = 0.9$ with $h = 10^{-k}$ for $k = 1, 2, 3....., 10$ using a first-order accurate forward difference approximation evaluated with f_{i+1} and f_i. Calculate the absolute error for each case using the exact value of the derivative at the given point. Comment on the results you obtain.

(*Hint: You may look at absolute error vs. h in a log-log plot for part (i) and part (ii), which may help you to explain the results.*)

Derivation of the formula above will be given Appendix A.

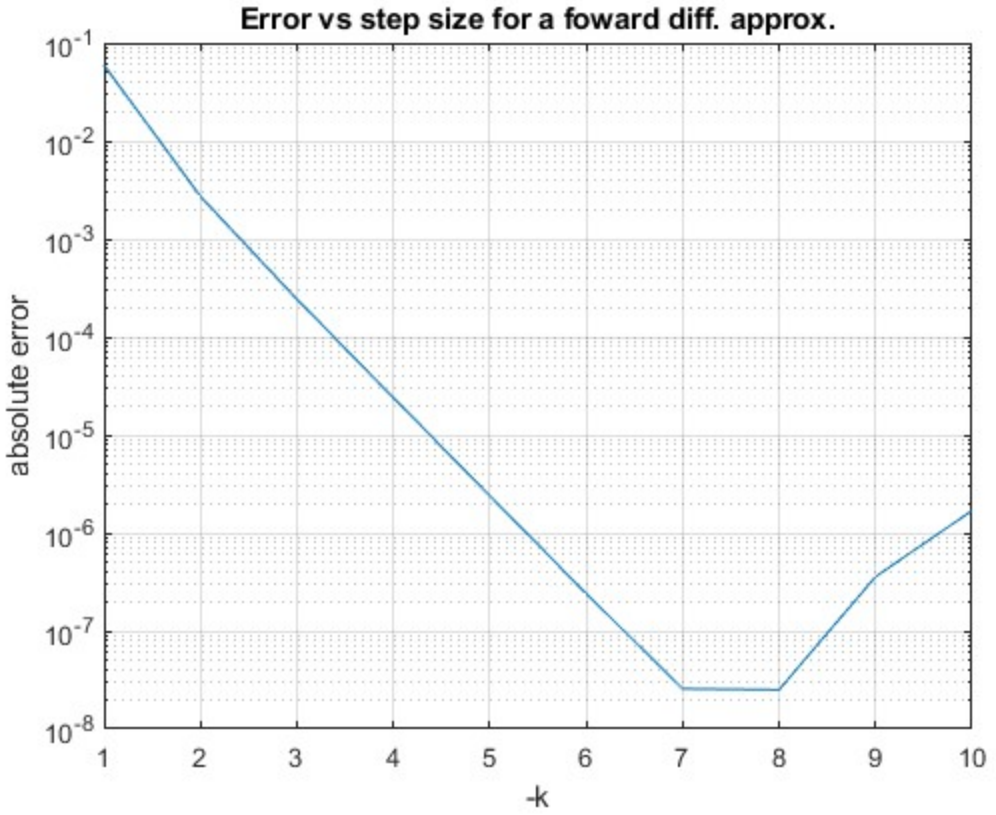


(i)

The step size reduces the error to a point, and then rises back up because the truncation error is overshadowed by the round-off error.

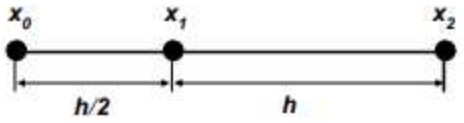
Tabulated derivatives and errors per method at a given step size x = 0.9

k	Cen. Err.	Cen. derivative	Fwd. Err.	Fwd. derivative
-1	3.86988467E-04	2.204753703	5.93714210E-02	2.144995293
-2	4.27651727E-08	2.204366757	2.71911483E-03	2.2016475995
-3	4.33342251E-12	2.204366714	2.42805930E-04	2.204123908
-4	1.74038561E-12	2.204366714	2.39925337E-05	2.204342722
-5	3.16324744E-12	2.204366714	2.39638270E-06	2.204364318
-6	3.36943806E-11	2.204366714	2.39941012E-07	2.204366474
-7	9.28498611E-10	2.204366713	2.57234798E-08	2.204366689
-8	1.50772164E-08	2.204366729	2.51683683E-08	2.204366689
-9	2.00953681E-07	2.204366513	3.58235276E-07	2.204366356
-10	7.14980315E-07	2.204367429	1.69050291E-06	2.204365024



A forward difference approximation behaves like the central difference method in that they reduce the error to some value of the step size but decreasing the step size increases the round-off error. The Central difference method is more accurate but requires more computing power.

Q4:



Using the non-uniform mesh spacing shown in the figure, derive

(a) a second order accurate one-sided finite difference formula to approximate (df/dx) at $x = x_0$.

(b) a first order accurate one-sided finite difference formula to approximate (d^2f/dx^2) at $x = x_0$.

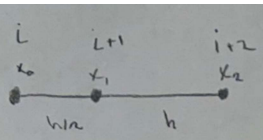
Show each step clearly in your derivations and give the leading term of the truncation error for each approximation.

a)

$$f_i' = \frac{f_{i+1} - f_i}{h} * \frac{f_{i+2} - f_{i+1} + 2f_i}{8h} - \frac{13h^2}{192} f_i'''$$

b)

$$f_i'' = \frac{f_{i+2} - 3f_{i+1} - f_i}{h^2} - \frac{13h}{24} f_i'''$$



$$f(x_0 + \tfrac{h}{2}) = f(x_0) + h/2\, f'(x_0) + \frac{h^2/4}{2!} f''(x_0) \cdots$$

$$f(x_0) = f(x_0) + f'(x_0) + \frac{1}{2!} f''(x_0)$$

$$f'(x_0) = \frac{f(x_0 + h/2) - f(x_0) + \frac{h^2/4}{2!} f''(x_0)}{h}$$

$$= \frac{f(x_0 + h/2) - f(x_0)}{h} + \frac{h}{8} f''(x_0)$$

$$f'(x_0) \approx \frac{f(x_0 + h/2) - f(x_0)}{h} + O(h)$$

$$f(x_0 + 3/2h) = f(x_0) + \tfrac{3}{2} h f'(x) + \frac{(3/2h)^2}{2!} f'' + \frac{(3/2h)^3}{3!} f'''(x_0)$$

$$f_{i+2} - 3 f_{i+1} = -2 f(x_0) + \frac{(3/2h)^2 - (h/2)^2}{2!} f'' + \frac{(3/2h)^3 - (3/2h)^3}{3} f''' + \cdots$$

$$f''_i = \left[\left(f_{i+2} - 3 f_{i+1} - f(x_0) - \frac{(3/2h)^3 - (h/2)^3}{3!} f''' \right) \left[\frac{2!}{(3/2h)^2 - (h/2)^2} \right] \right] \cdots$$

$$= \left[f(x_0 + \tfrac{3}{2}h) - 3 f(x_0 + h/2) - f(x_0) - \frac{13h^3}{24} f''' \right] / h^2$$

b)

$$\approx \frac{f(x_0 + \frac{3}{2}h) - 3 f(x_0 + \frac{h}{2}) - f(x_0)}{h^2} - \frac{13h}{24} f'''(x_0)$$

$$f'(x_0) = \frac{f(x_0 + \frac{h}{2}) - f(x_0)}{h} + \frac{h}{8} f''(x_0) \quad (1)$$

$f''(x_0) \quad O(h^2)$

$$f_{i+2} - 3f_{i+1} = -2f(x_0) + \frac{(3/2h)^2 - (h/2)^2}{2!} f_i'' + \frac{(3/2h)^3 - (h/2)^3}{3!} f_i''' + \frac{(3/2h)^4 - (h/2)^4}{4!} f_i^{iv}$$

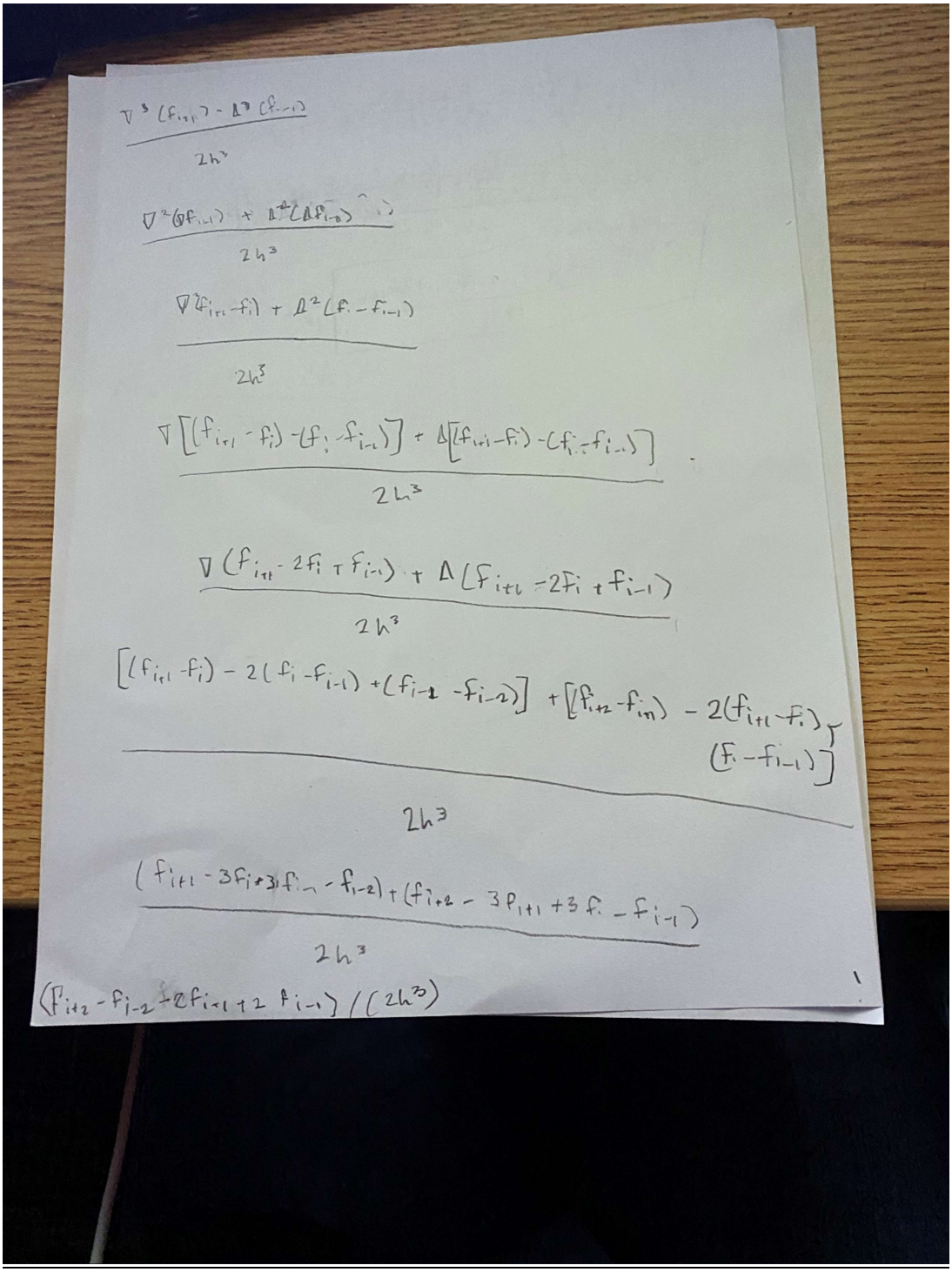
$$f_i'' = \frac{(f_{i+2} - f_{i+1}) + 2f_i - \frac{13h^3}{24} f_i''' - \frac{5h^4}{1} f^{iv}}{h^2}$$

$$f_i'' = \frac{f_{i+2} - f_{i+1} + 2f_i}{h^2} - \frac{13h}{24} f''' - 5h^2 f^{iv} \quad (2)$$

Sub (2) in (1)

$$f_i' = \underbrace{\frac{f_{i+1} - f_i}{h} + \frac{f_{i+2} - f_{i-1} + 2f_i}{8h}}_{\text{Approx}} \underbrace{- \frac{13h^2}{42} f''' - \frac{5}{8} h^3 f_i^{iv}}_{\text{True error}}$$

Appendix A – q3 derivation



$$\frac{\nabla^3(f_{i+1}) - \Delta^3(f_{i-1})}{2h^3}$$

$$\frac{\nabla^2(\nabla f_{i+1}) + \Delta^2(\Delta f_{i-1})}{2h^3}$$

$$\frac{\nabla^2(f_{i+1} - f_i) + \Delta^2(f_i - f_{i-1})}{2h^3}$$

$$\frac{\nabla[(f_{i+1} - f_i) - (f_i - f_{i-1})] + \Delta[(f_{i+1} - f_i) - (f_i - f_{i-1})]}{2h^3}$$

$$\frac{\nabla(f_{i+1} - 2f_i + f_{i-1}) + \Delta(f_{i+1} - 2f_i + f_{i-1})}{2h^3}$$

$$\frac{[(f_{i+1} - f_i) - 2(f_i - f_{i-1}) + (f_{i-1} - f_{i-2})] + [(f_{i+2} - f_{i+1}) - 2(f_{i+1} - f_i) (f_i - f_{i-1})]}{2h^3}$$

$$\frac{(f_{i+1} - 3f_i + 3f_{i-1} - f_{i-2}) + (f_{i+2} - 3f_{i+1} + 3f_i - f_{i-1})}{2h^3}$$

$$(f_{i+2} - f_{i-2} - 2f_{i+1} + 2f_{i-1}) / (2h^3)$$

$$(f_i + hf_i' + \frac{h^2}{2!} f_i'' + \frac{h^3}{3!} f_i''' + \frac{h^4}{4!} f_i'''' + \frac{h^5}{5!} f_i^{v})$$

$$- (f_i - hf_i' + \frac{h^2}{2!} f_i'' - \frac{h^3}{3!} f_i''' + \frac{h^4}{4!} f_i'''' - \frac{h^5}{5!} f_i^{v})$$

$$2hf_i' \qquad + 2\frac{h^3}{3!} f_i''' \qquad + 2\frac{h^5}{5!} f^{v}$$

$$f_{i+1} - f_{i-1} = 2hf_i' + 2\frac{h^3}{3!} f''' \ldots$$

$$f_i' = \frac{f_{i+1} - f_{i-1} - (2\frac{h^3}{3!}) f'''}{2h}$$

$$f_{i+1} - f_{i-1} - \left[\frac{h^3}{3}\left(\frac{f_{i+2} - f_{i-2}}{2h^3}\right) + 2\left(f_{i+1} - f_{i-1}\right)\right]$$

$$\frac{6f_{i+1} - 6f_{i-1} - \left[\frac{f_{i+2} - f_{i-2} + 8f_{i+1} - f_{i-1}}{6}\right]}{2h}$$

$$f_i' = \frac{-f_{i+2} + f_{i-2} - 8f_{i-1} + 8f_{i+1}}{12h}$$

Appendix B – code

```
% main.m
clc
clear all
close all
format longg

%=========================================================================%
                        % q1
%=========================================================================%
syms xx
interval = 2;
x = transpose(0:2/interval:2);
f = @(x) sin(exp(x)-2);
y = f(x);
n = interval;
fapprox3(xx) = Lagran(x,y,n,xx);
interval = 5;
x = transpose(0:2/interval:2);
y = f(x);
n = interval;
fapprox6(xx) = Lagran(x,y,n,xx);
figure(1);
hold on
fplot(f)
fplot(fapprox3)
fplot(fapprox6)
xlim([ 0 2 ])
ylim([-1 1])
title('Langrange interpolation')
xlabel('x')
ylabel('f(x)')
legend('actual','3rd degree','6th degree','Location','northwest')
grid on
hold off
x = [.1;.9;1.5;1.9];
err3 = double(f(x)-fapprox3(x));
err6 = double(f(x)-fapprox6(x));
err = [x err3 err6];
figure(2);
hold on
fplot((f(xx)-fapprox3(xx)))
fplot((f(xx)-fapprox6(xx)))
xlim([ 0 2 ])
ylim([-1 1])
title('Langrange interpolation error')
xlabel('x')
ylabel('f(x)')
legend('3rd degree','6th degree','Location','northwest')
grid on
hold off
```

```
%=========================================================================%
                        % q2
%=========================================================================%
syms p
a = 0; % lower bound
n = 11;
b = 2; % upper bound
points = 11;
x = transpose(a:(b-a)/(points):b);
fun = @(x) sin(exp(x)-2);
g = fun(x);
[f] = NCSpline.sysMake(g,n,x);
spline = sym(zeros(b*100,1));
err = spline;
i = 0 ;
j=1;
while(true)
    if i>=0 && i<b/n
        spline(j) = subs(f(1),p,i);
    end
    if i<2*b/n && i>=b/n
        spline(j) = subs(f(2),p,i);
    end
    if i<3*b/n &&i>=2*b/n
        spline(j) = subs(f(3),p,i);
    end
    if i<4*b/n &&i>=3*b/n
        spline(j) = subs(f(4),p,i);
    end
    if i<5*b/n &&i>=4*b/n
        spline(j) = subs(f(5),p,i);
    end
    if i<6*b/n &&i>=5*b/n
        spline(j) = subs(f(6),p,i);
    end
    if i<7*b/n &&i>=6*b/n
        spline(j) = subs(f(7),p,i);
    end
    if i<8*b/n &&i>=7*b/n
        spline(j) = subs(f(8),p,i);
    end
    if i<9*b/n &&i>=8*b/n
        spline(j) = subs(f(9),p,i);
    end
    if i<10*b/n &&i>=9*b/n
        spline(j) = subs(f(10),p,i);
    end
    if i<11.1*b/n &&i>=10*b/n
        spline(j) = subs(f(11),p,i);
    end
    err(j,1) = abs(fun(i)-spline(j));
    i=i+.01;
    j =j+1;
    if i>=b
        break;
    end
    spline = double(spline);
```

```
    err = double(err)

    hold on
    fplot(fun)
    plot((1:length(spline))/100,spline)
    plot((1:length(spline))/100,err)
    legend('actual','approximate','error','Location','northwest')
    title('Comparison of spline and function')
    grid on
    xlabel('x')
    ylabel('f(x)')
    xlim([a b])
    hold off

end
%=========================================================================%
                        % q3
%=========================================================================%

x = 0.9;
fpEx = fprimeEx(x)*ones(10,1);
fp = zeros(10,1);
fwd_fp = zeros(10,1);
for k = 1:10
     h = 10^-k;
     fp2 = f(x+2*h);
     fp1 = f(x+h);
     fm1 = f(x-h);
     fm2 = f(x-2*h);
     fp(k,1) = fprime(fp2,fp1,fm1,fm2,h);
     fwd_fp(k,1) = fwd_fprime(fp1,f(x),h);
end

err = abs(fp - fpEx)
fwd_err = abs(fwd_fp - fpEx)

semilogy(1:10,err)
title('Error vs step size for a central diff. approx.')
xlabel('-k')
ylabel('absolute error')
grid on
semilogy(1:10,fwd_err)
title('Error vs step size for a foward diff. approx.')
xlabel('-k')
ylabel('absolute error')
grid on

function y = f(x)
    y = sin(exp(x)-2);
end
function y = fprimeEx(x)
    y = exp(x)*cos(exp(x)-2);
end
```

```
function f = Lagran(x,y,n,xx)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here


sum = 0;
for i = 1:n+1
    product = y(i,1);
    for j = 1:n+1
        if i~=j
            product = product*(xx-x(j,1))/(x(i,1)-x(j,1));
        end
    end
    sum = sum + product;
end
f = sum;
end

function fp = fprime(fp2,fp1,fm1,fm2,h)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
fp = 1/12/h*(-fp2+8*fp1-8*fm1+fm2);
end

function [fp] = fwd_fprime(fp1,fi,h)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
fp = (fp1-fi)/h;
end


classdef NCSpline
    %UNTITLED Summary of this class goes here
    %   Detailed explanation goes here

    methods (Static)
        function [s] = sysMake(g,n,x)
            %sysMake will Construct a system for NC spline
            %   it will find the coefficents of each spline and will output
            %   a column vector of the spline equations as syms
            % g is the vector of known points, x is the vector coresponding
            % to the points
            % n is the number of intervals inbetween all points
            syms p
            h = zeros(n,1);
            for i = 1:n-1
                h(i,1) = x(i+1)-x(i);
            end
            A = zeros(n);
            A(1,1) = 1;
            A(n,n) = 1;
            for i = 2:n-1
                A(i,i) = 2*(h(i-1)+h(i));
            end
            for i = 1:n-1
```

```
                A(i+1,i) = h(i);
                A(i,i+1) = h(i);
            end
            A(1,2) = 0;
            A(n,n-1) = 0 ;
            b = zeros(n,1);
            for i = 2:n-1
                b(i,1) = 3/h(i)*(g(i+1)-g(i))-3/h(n-1)*(g(i)-g(i-1));
            end
            c = gauss(A,b);
            k = zeros(n,1);
            d = zeros(n,1);
            for i = 1:n-1
                k(i,1) = (g(i+1)-g(i))/h(i)-h(i)/3*(2*c(i)+c(i+1));
                d(i,1) = (c(i+1)-c(i))/3/h(i);
            end
            s = sym(zeros(n,1));
            for i = 1 :n
                s(i,1) = (g(i)+k(i)*(p-x(i))+c(i)*(p-x(i))^2+d(i)*(p-
x(i))^3);
            end
        end


    end
end



function [x] = gauss(a,b)
% gauss elimination

n = length(a);

k = 1 ;
p = k ;
big = abs(a(k,k));

%**************************************************
% pivoting portion
%**************************************************
for ii=k+1:n
    dummy = abs(a(ii,k));
    if dummy > big
       big = dummy;
       p = ii ;
    end
end
if p ~= k
    for jj = k:n
        dummy = a(p,jj);
        a(p,jj) = a(k,jj);
        a(k,jj) = dummy;
    end
    dummy = b(p);
```

```
    b(p)=b(k);
    b(k) = dummy;
end

%*************************************************
% elimination step
%*************************************************
for k=1:(n-1)
    for i=k+1:n
        factor = a(i,k)/a(k,k);
       for j=k+1:n
           a(i,j) = a(i,j) - factor*a(k,j);
       end
       b(i) = b(i) - factor*b(k);
    end
end

%*************************************************
% back substitution
%*************************************************
x(n,1) = b(n)/a(n,n);
for i = n-1:-1:1
    sum = b(i);
   for j = i + 1:n
       sum = sum - a(i,j)*x(j,1);
   end
   x(i,1) = sum/a(i,i);
end
end
```