

# CS / MATH 4334 : Numerical Analysis

## Homework Assignment 4

Matthew McMillian  
mgm160130@utdallas.edu

October 31, 2018

### **MatLab Problems**

```

1  format long e
2
3  clc
4
5  nnodes = 10;
6
7  a = -3;
8  b = 3;
9
10 % Creates evenly spaced nnodes
11 %iter = (abs(a) + abs(b)) / (nnodes-1);
12 %tmp = a;
13
14 % Fills out nodes
15 % nodes(1) = a;
16 % for i = 2:nnodes-1
17 % tmp = tmp + iter;
18 % nodes(i) = tmp;
19 % end
20 % nodes(nnodes) = b;
21
22 % Fills out nodes
23 nodesx = linspace(a,b,nnodes);
24 nodesy = exp(-1*abs(nodesx));
25
26 % Evalute function at points
27 points = 600; % 0.01 spaced points
28
29 x = linspace(a,b, 600);
30 y = exp(-abs(x));
31
32 % Computing the coefs
33 coefs = newtdd(nodesx,nodesy,nnodes);
34
35 % Computing the polynomial
36 poly = nest(nnodes-1, coefs, x, nodesx);
37
38 % Error
39 abserr = max(abs(y - poly))
40
41 % Plotting
42 figure()

```

```

43 plot(x ,y)
44 hold on
45 plot(x, poly)
46 hold off
47 s = sprintf("Actual vs Interpolated f(x) at %d nodes", nnodes)
    ;
48 title(s)
49 legend({ 'regular', 'interpolated'}, 'Location','southwest')
50
51 nnodes = 30;
52
53 a = -3;
54 b = 3;
55
56 % Creates evenly spaced nnodes
57 %iter = (abs(a) + abs(b)) / (nnodes-1);
58 %tmp = a;
59
60 % Fills out nodes
61 % nodes(1) = a;
62 % for i = 2:nnodes-1
63 % tmp = tmp + iter;
64 % nodes(i) = tmp;
65 % end
66 % nodes(nnodes) = b;
67
68 % Fills out nodes
69 nodesx = linspace(a,b,nnodes);
70 nodesy = exp(-1*abs(nodesx));
71
72 % Evaluate function at points
73 points = 600; % 0.01 spaced points
74
75 x = linspace(a,b, 600);
76 y = exp(-abs(x));
77
78 % Computing the coefs
79 coefs = newtdd(nodesx ,nodesy ,nnodes);
80
81 % Computing the polynomial
82 poly = nest(nnodes-1, coefs , x, nodesx);
83
84 % Error
85 abserr = max(abs(y - poly))
86

```

```

87 % Plotting
88 figure()
89 plot(x ,y)
90 hold on
91 plot(x, poly)
92 hold off
93 ylim([0,1])
94 s = sprintf("Actual vs Interpolated f(x) at %d nodes", nnodes)
95     ;
96 title(s)
97 legend({'regular', 'interpolated'}, 'Location','southwest')

```

Problem 1: plchev.m \_\_\_\_\_

```

1 format long e
2
3 clc
4
5 nnodes = 10;
6
7 a = -3;
8 b = 3;
9
10 nodesx = (nnodes);
11
12 % Creates evenly spaced nnodes
13 iter = (abs(a) + abs(b)) / (nnodes-1);
14 tmp = a;
15
16 % Fills out nodes
17 for i = 1:nnodes
18     nodesx(i) = ((b+a)/2) + ((b-a)/2)*cos((2*i - 1)*pi)/(2*
19         nnodes));
20
21
22 nodesy = exp(-1*abs(nodesx));
23
24 % Evalute function at points
25 points = 600; % 0.01 spaced points
26
27 x = linspace(a,b, 600);
28 y = exp(-abs(x));
29

```

```

30 % Computing the coefs
31 coefs = newtdd(nodesx, nodesy, nnodes);
32
33 % Computing the polynomial
34 poly = nest(nnodes-1, coefs, x, nodesx);
35
36 % Error
37 abserr = max(abs(y - poly))
38
39 % Plotting
40 figure()
41 plot(x, y)
42 hold on
43 plot(x, poly)
44 hold off
45 s = sprintf("Actual vs Interpolated f(x) at %d nodes", nnodes)
    ;
46 title(s)
47 legend({'regular', 'interpolated'}, 'Location', 'southwest')
48
49
50
51
52
53 nnodes = 30;
54
55 a = -3;
56 b = 3;
57
58 nodesx = (nnodes);
59
60 % Creates evenly spaced nnodes
61 iter = (abs(a) + abs(b)) / (nnodes-1);
62 tmp = a;
63
64 % Fills out nodes
65 for i = 1:nnodes
66     nodesx(i) = ((b+a)/2) + ((b-a)/2)*cos((2*i - 1)*pi)/(2*
        nnodes));
67 end
68
69
70 nodesy = exp(-1*abs(nodesx));
71
72 % Evalute function at points

```

```

73 points = 600; % 0.01 spaced points
74
75 x = linspace(a,b, 600);
76 y = exp(-abs(x));
77
78 % Computing the coefs
79 coefs = newtdd(nodesx,nodesy,nnodes);
80
81 % Computing the polynomial
82 poly = nest(nnodes-1, coefs, x, nodesx);
83
84 % Error
85 abserr = max(abs(y - poly))
86
87 % Plotting
88 figure()
89 plot(x,y)
90 hold on
91 plot(x, poly)
92 hold off
93 s = sprintf("Actual vs Interpolated f(x) at %d nodes", nnodes)
94 ;
95 title(s)
96 legend({'regular', 'interpolated'}, 'Location','southwest')

```

```
>> p1.m
```

```
abserr =
```

```
2.142188221906679e-01
```

```
abserr =
```

```
1.894862537027103e+03
```

```
>> p1chev.m
```

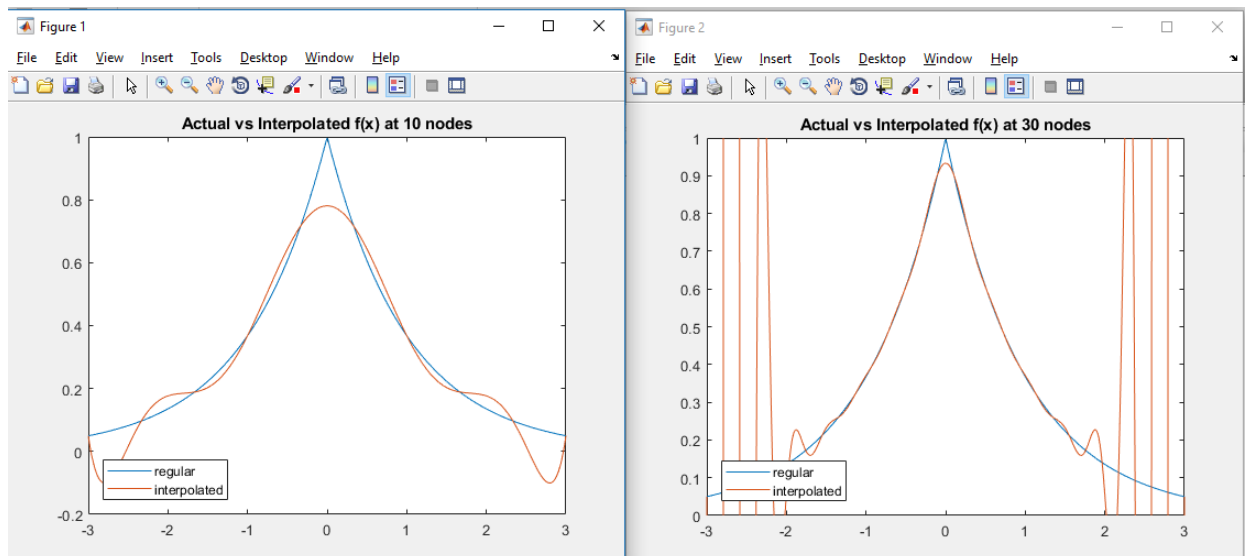
```
abserr =
```

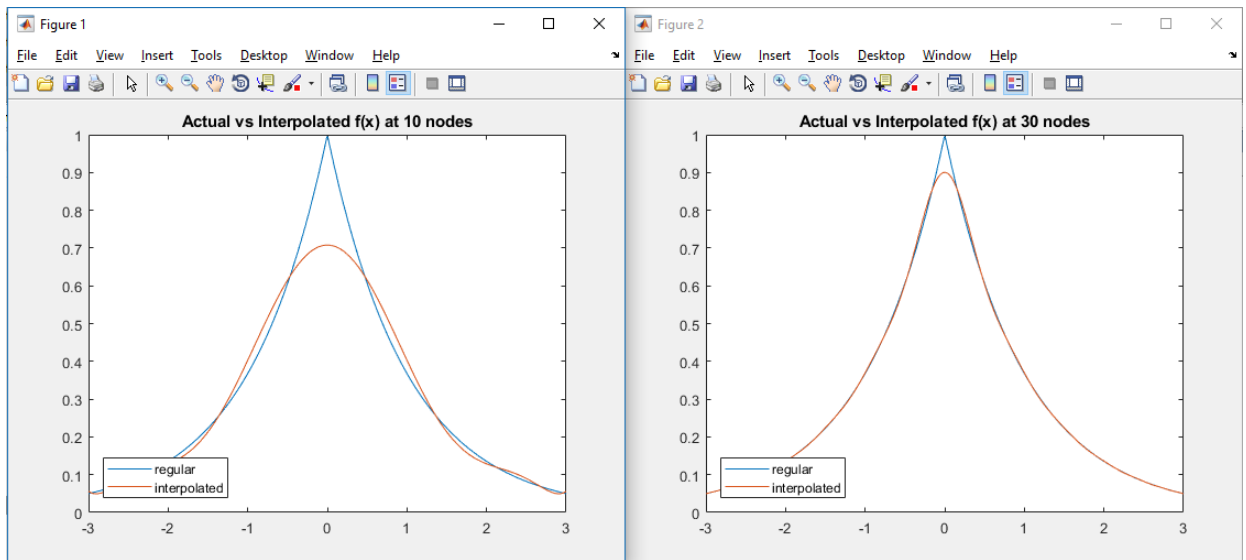
```
2.872137290339052e-01
```

```
abserr =
```

```
9.477703742070265e-02
```

You can compute the error bounds of both (a) and (b) since you have access to the initial function. If you only had data points, you could not compute the theoretical error.







## Problem 2 : quadspline.m

---

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% compute spline coefficients here
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2
3  format long e
4
5  clc
6
7  a = 0;
8  b = 2*pi;
9
10 % Num of nodes + array
11 n = 100;
12 y = (n);
13 x = (n);
14
15 % A matrix creation for 100 nodes
16 A = zeros(n-2, n-2);
17
18 A(1,1) = 1;
19 for i=2:n-1
20     A(i, i) = 1;
21     A(i, i-1) = 1;
22 end
23 A(n-2, n-2) = 1;
24 A(n-2, n-1) = 1;
25
26
27 % Spacing of nodes
28 iter = (abs(a) + abs(b)) / (n-1);
29 tmp = a;
30
31 % Setting up x and y values
32 x = linspace(0,2*pi);
33 y = cos(x);
34
35 % Setup a
36 a = zeros(n-1,1);
37 for i=1:n-1
38     a(i) = y(i);
39 end
40
41 fprintf("Det(A) = 1\n")

```

```

42
43 % Setup Solution Vector b
44 b = zeros(n-2,1);
45 b(1) = 0;
46 for i = 2:n-1
47     %b(i) = ((2/(x(i+1)-x(i)))*(y(i+1)-y(i))) - b(i-1);
48     b(i) = ((2/(x(i)-x(i-1)))*(y(i)-y(i-1))));
49 end
50 b = forsub(A,b);
51
52 % Setup c
53 c = zeros(n-1,1);
54 for i=1:n-1
55     c(i) = ((y(i+1) - y(i))/(x(i+1) - x(i))^2) - (b(i)/(x(i
        +1) - x(i)));
56 end
57
58
59 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% call sineval function here
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
60
61 X = 250;
62
63 cosine(X,a,b,c)
64
65 X = -100;
66
67 cosine(X,a,b,c)
68
69
70 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% plot spline
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
71
72 %number of points on which to plot. n = number of nodes
73 nplot = (n-1)*19+1;
74
75 xplot = zeros(nplot,1);
76 yplot = zeros(nplot,1);
77
78 %spacing between plot points
79 nspace = (x(n)-x(1))/(nplot-1);
80
81 k = 0;
82 for i = 1:n-1
83

```

```

84     for j = 1:19
85
86         k = k+1;
87         xplot(k) = x(i) + (j-1)*nspace;
88         yplot(k) = a(i) + b(i)*(xplot(k) - x(i)) + c(i)*(xplot
            (k) - x(i))^2;
89
90     end
91
92 end
93
94 xplot(nplot) = x(n);
95 yplot(nplot) = a(i) + b(i)*(x(n) - x(n-1)) + c(i)*(x(n) - x(n
    -1))^2;
96
97 plot(xplot, yplot)
98
99 figure()
100
101 abserr = abs(yplot - cos(xplot));
102
103 plot(xplot, abserr)

```

Problem 2: cosine.m

---

```

1  function [approx] = cosine(X, a, b, c)
2  %COSINE Summary of this function goes here
3  % Detailed explanation goes here
4
5  X = mod(abs(X), 2*pi);
6
7  k = floor(X/(2*pi/99));
8
9  approx = a(k) + b(k)*(X - (k-1)*(2*pi/99)) + c(k)*(X - (k-1)
    *(2*pi/99))^2;

```

```
>> quadspline.m
```

Example System for n=5:

1 0 0 0—b1—v1

1 1 0 0—b2— $2/\delta * (y_3 - y_2) - v_1$

0 1 1 0—b3— $2/\delta * (y_4 - y_3) - 2/\delta * (y_3 - y_2)$

0 0 1 1—b4— $2/\delta * (y_5 - y_4) - 2/\delta * (y_4 - y_3)$

$\text{Det}(A) = 1$

ans =

2.409905359662697e-01

ans =

8.623460672974910e-01

