CS / MATH 4334 : Numerical Analysis
Homework Assignment 4

Matthew McMillian
mgm160130@utdallas.edu

October 31, 2018

MatLab Problems

Problem 1 : p1.m

```
format long e

clc

nnodes = 10;

a = -3;
b = 3;

% Creates evenly spaced nnodes
%iter = (abs(a) + abs(b)) / (nnodes-1);
%tmp = a;

% Fills out nodes
% nodes(1) = a;
% for i = 2:nnodes-1
% tmp = tmp + iter;
% nodes(i) = tmp;
% end
% nodes(nnodes) = b;

% Fills out nodes
nodesx = linspace(a,b,nnodes);
nodesy = exp(-1*abs(nodesx));

% Evalute function at points
points = 600; % 0.01 spaced points

x = linspace(a,b, 600);
y = exp(-abs(x));

% Computing the coefs
coefs = newtdd(nodesx,nodesy,nnodes);

% Computing the polynomial
poly = nest(nnodes-1, coefs, x, nodesx);

% Error
abserr = max(abs(y - poly))

% Plotting
figure()
```

```
plot(x  ,y)
hold on
plot(x, poly)
hold off
s = sprintf("Actual vs Interpolated f(x) at %d nodes", nnodes)
    ;
title(s)
legend({'regular', 'interpolated'}, 'Location','southwest')

nnodes = 30;

a = -3;
b = 3;

% Creates evenly spaced nnodes
%iter = (abs(a) + abs(b)) / (nnodes-1);
%tmp = a;

% Fills out nodes
% nodes(1) = a;
% for i = 2:nnodes-1
% tmp = tmp + iter;
% nodes(i) = tmp;
% end
% nodes(nnodes) = b;

% Fills out nodes
nodesx = linspace(a,b,nnodes);
nodesy = exp(-1*abs(nodesx));

% Evalute function at points
points = 600; % 0.01 spaced points

x = linspace(a,b, 600);
y = exp(-abs(x));

% Computing the coefs
coefs = newtdd(nodesx,nodesy,nnodes);

% Computing the polynomial
poly = nest(nnodes-1, coefs , x, nodesx);

% Error
abserr = max(abs(y - poly))
```

```
% Plotting
figure()
plot(x ,y)
hold on
plot(x, poly)
hold off
ylim([0,1])
s = sprintf("Actual vs Interpolated f(x) at %d nodes", nnodes);
title(s)
legend({'regular', 'interpolated'}, 'Location','southwest')
```

Problem 1 : p1chev.m

```
format long e

clc

nnodes = 10;

a = -3;
b = 3;

nodesx = (nnodes);

% Creates evenly spaced nnodes
iter = (abs(a) + abs(b)) / (nnodes-1);
tmp = a;

% Fills out nodes
for i = 1:nnodes
    nodesx(i) = ((b+a)/2) + ((b-a)/2)*cos(((2*i - 1)*pi)/(2*nnodes));
end


nodesy = exp(-1*abs(nodesx));

% Evalute function at points
points = 600; % 0.01 spaced points

x = linspace(a,b, 600);
y = exp(-abs(x));

```

```
% Computing the coefs
coefs = newtdd(nodesx,nodesy,nnodes);

% Computing the polynomial
poly = nest(nnodes-1, coefs, x, nodesx);

% Error
abserr = max(abs(y - poly))

% Plotting
figure()
plot(x ,y)
hold on
plot(x, poly)
hold off
s = sprintf("Actual vs Interpolated f(x) at %d nodes", nnodes)
    ;
title(s)
legend({'regular', 'interpolated'}, 'Location','southwest')





nnodes = 30;

a = -3;
b = 3;

nodesx = (nnodes);

% Creates evenly spaced nnodes
iter = (abs(a) + abs(b)) / (nnodes-1);
tmp = a;

% Fills out nodes
for i = 1:nnodes
    nodesx(i) = ((b+a)/2) + ((b-a)/2)*cos(((2*i - 1)*pi)/(2*
        nnodes));
end


nodesy = exp(-1*abs(nodesx));

% Evalute function at points
```

```
points = 600; % 0.01 spaced points

x = linspace(a,b, 600);
y = exp(-abs(x));

% Computing the coefs
coefs = newtdd(nodesx,nodesy,nnodes);

% Computing the polynomial
poly = nest(nnodes-1, coefs, x, nodesx);

% Error
abserr = max(abs(y - poly))

% Plotting
figure()
plot(x ,y)
hold on
plot(x, poly)
hold off
s = sprintf("Actual vs Interpolated f(x) at %d nodes", nnodes)
    ;
title(s)
legend({'regular', 'interpolated'}, 'Location','southwest')
```

>> p1.m

```
abserr =

2.142188221906679e-01

abserr =

1.894862537027103e+03
```

>> p1chev.m

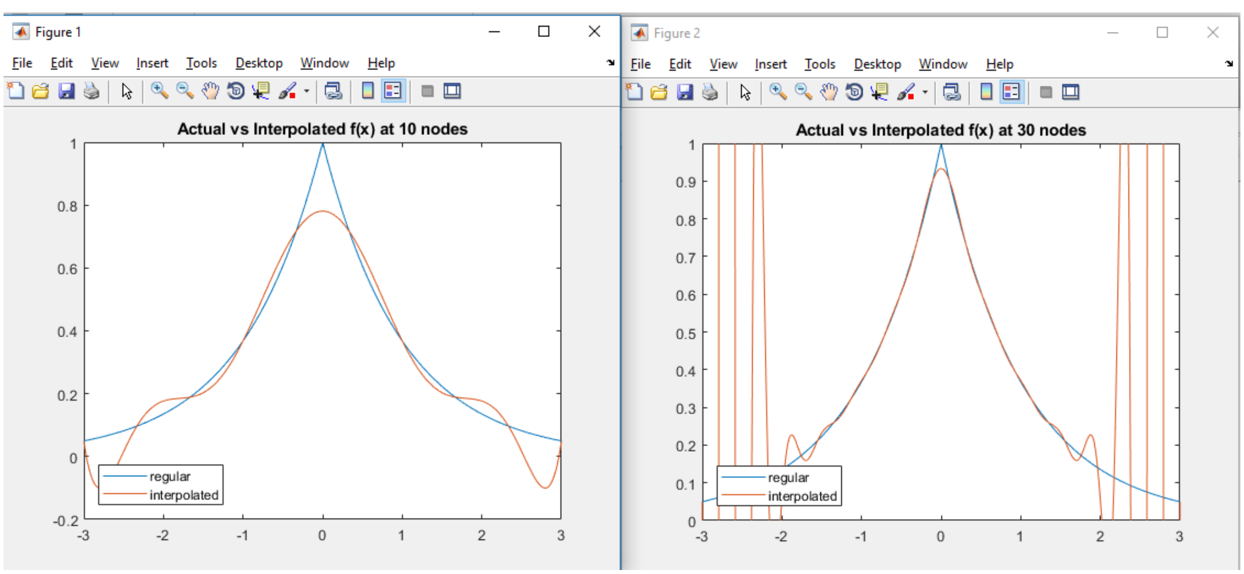
```
abserr =

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.



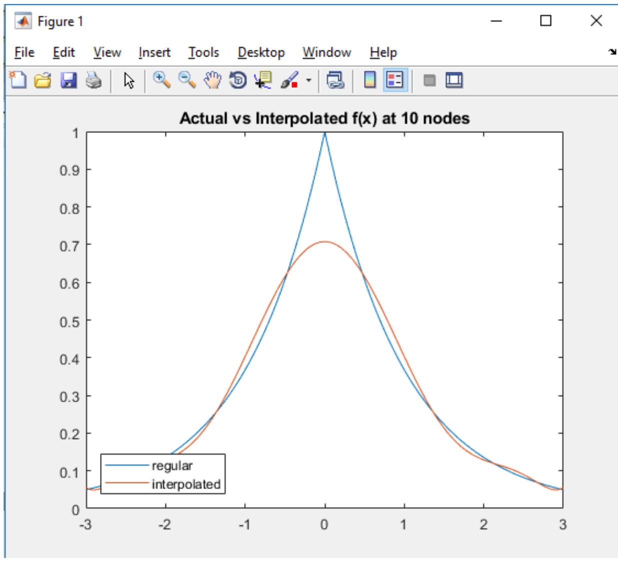


Figure 1
File Edit View Insert Tools Desktop Window Help
Actual vs Interpolated f(x) at 10 nodes
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
-3 -2 -1 0 1 2 3
regular
interpolated

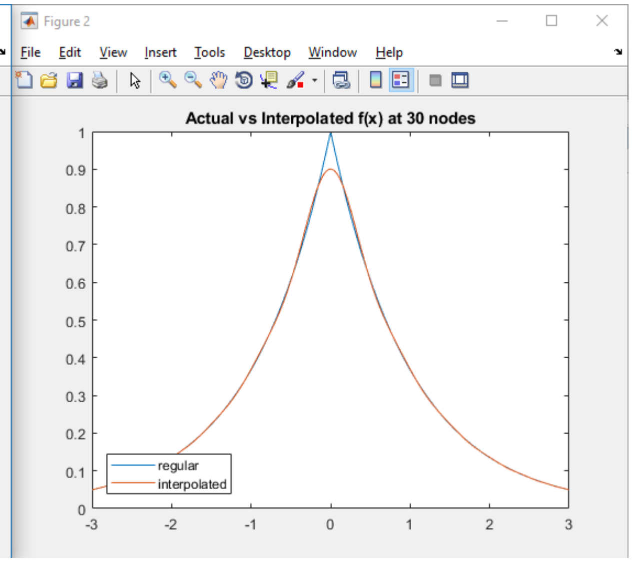


Figure 2
File Edit View Insert Tools Desktop Window Help
Actual vs Interpolated f(x) at 30 nodes
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
-3 -2 -1 0 1 2 3
regular
interpolated

Problem 2 : quadspline.m

```
%%%%%%%%%%%%%%%%%%%%%      compute spline coefficients here
    %%%%%%%%%%%%%%%

format long e

clc

a = 0;
b = 2*pi;

% Num of nodes + array
n = 100;
y = (n);
x = (n);

% A matrix creation for 100 nodes
A = zeros(n-2, n-2);

A(1,1) = 1;
for i=2:n-1
     A(i, i) = 1;
     A(i, i-1) = 1;
end
A(n-2, n-2) = 1;
A(n-2, n-1) = 1;


% Spacing of nodes
iter = (abs(a) + abs(b)) / (n-1);
tmp = a;

% Setting up x and y values
x = linspace(0,2*pi);
y = cos(x);

% Setup a
a = zeros(n-1,1);
for i=1:n-1
     a(i) = y(i);
end

fprintf("Det(A) = 1\n")
```

```
% Setup Solution Vector b
b = zeros(n-2,1);
b(1) = 0;
for i = 2:n-1
   %b(i) = ((2/(x(i+1)-x(i)))*(y(i+1)-y(i))) - b(i-1);
    b(i) = ((2/(x(i)-x(i-1)))*(y(i)-y(i-1)));
end
b = forsub(A,b);

% Setup c
c = zeros(n-1,1);
for i=1:n-1
    c(i) = ((y(i+1) - y(i))/((x(i+1) - x(i))^2)) - (b(i)/(x(i+1) - x(i)));
end


%%%%%%%%%%%%%%%%%% call sineval function here %%%%%%%%%%%%%%%%%%

X = 250;

cosine(X,a,b,c)

X = -100;

cosine(X,a,b,c)


%%%%%%%%%%%%%%%%%%%%%%% plot spline %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%number of points on which to plot. n = number of nodes
nplot  = (n-1)*19+1;

xplot = zeros(nplot,1);
yplot = zeros(nplot,1);

%spacing between plot points
nspace = (x(n)-x(1))/(nplot-1);

k = 0;
for i = 1:n-1

```

```
    for j = 1:19

        k = k+1;
        xplot(k) = x(i) + (j-1)*nspace;
        yplot(k) = a(i) + b(i)*(xplot(k) - x(i)) + c(i)*(xplot
            (k) - x(i))^2;

    end

end

xplot(nplot) = x(n);
yplot(nplot) = a(i) + b(i)*(x(n) - x(n-1)) + c(i)*(x(n) - x(n
    -1))^2;

plot(xplot,yplot)

figure()

abserr = abs(yplot - cos(xplot));

plot(xplot,abserr)
```

Problem 2 : cosine.m

```
function [approx] = cosine(X, a, b, c)
%COSINE Summary of this function goes here
%   Detailed explanation goes here

X = mod(abs(X), 2*pi);

k = floor(X/(2*pi/99));

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 * (y3-y2) - v1
0 1 1 0–*–b3–=—2/delta * (y4-y3) - 2/delta * (y3-y2)
0 0 1 1——b4——2/delta * (y5-y4) - 2/delta * (y4-y3)

Det(A) = 1

ans =

2.409905359662697e-01

ans =

8.623460672974910e-01