Lab 04

Table of Contents

Problem 1

Approximate f(0.8), f(1.2), and f(1.7) given the data in the table

```
%       x   | f(x)
%       ----------------
%       0.5 | 1.772454
%       0.7 | 1.298055
%       1.0 | 1.000000
%       1.3 | 0.897471
%       1.5 | 0.886227
%       1.6 | 0.893515
%       2.0 | 1.000000

fprintf('Problem 1\n');

% for x = 0.8, I will use the following indexing array to rearrange
 the
%   data points based on their proximity to 0.8
x = 0.8;
indexingArray = [2, 3, 1, 4, 5, 6, 7]; % 0.7, 1.0, 0.5, 1.0, 1.3, 1.6,
 2.0
approximation = GetAnswerP1(x, indexingArray);
fprintf('The approximate value of f(%.1f) = %.6f\n', x, approximation)

% for x = 1.2, I will use the following indexing array to rearrange
 the
%   data points based on their proximity to 1.2
x = 1.2;
indexingArray = [4, 3, 5, 6, 2, 1, 7]; % 1.3, 1.0, 1.5, 1.6, 0.7, 0.5,
 2.0
approximation = GetAnswerP1(x, indexingArray);
fprintf('The approximate value of f(%.1f) = %.6f\n', x, approximation)

% for x = 1.7, I will use the following indexing array to rearrange
 the
```

```
%   data points based on their proximity to 1.2
x = 1.7;
indexingArray = [6, 5, 7, 4, 3, 2, 1]; % 1.6, 1.5, 2.0, 1.3, 1.0, 0.7,
 0.5
approximation = GetAnswerP1(x, indexingArray);
fprintf('The approximate value of f(%.1f) = %.6f\n', x, approximation)
```

```
Problem 1
```

Problem 2

Generate function data for f(x) = 1 / (1 + (x^2)) on N equally-spaced nodes on the interval [-5,5]. Use Neville's method to approximate f(4.9) for N in {11, 21, 41, 81, 121, 161}.

```
fprintf('Problem 2\n');
x = 4.9; % used for all values of n

spacing = [11, 21, 41, 81, 121, 161]; % vector containing spacings
for n = 1: length(spacing)
    approximation = GetAnswerP2and3(spacing(n), x, 2);
    fprintf('N: %d \tApproximation: %.6f\n', spacing(n),
 approximation);
end
```

```
Problem 2
N: 11  Approximation: 1.230317
N: 21  Approximation: -58.238141
N: 41  Approximation: -78688.997501
N: 81  Approximation: -40443044569.410362
N: 121  Approximation: 35235652991531112.000000
N: 161  Approximation: 44255285491083409063149568.000000
```

Problem 3

Repeat Problem 2 with function data at the Chebyshev nodes defined by x = -5 * cos(((2k-1) / (2*N))*pi), k in 1..N

```
fprintf('Problem 3\n');
x = 4.9; % used for all values of n

spacing = [11, 21, 41, 81, 121, 161]; % vector containing spacings
for n = 1: length(spacing)
    approximation = GetAnswerP2and3(spacing(n), x, 3);
    fprintf('N: %d \tApproximation: %.12f\n', spacing(n),
 approximation);
end
```

```
Problem 3
N: 11  Approximation: 0.066370484725
N: 21  Approximation: 0.037059326736
N: 41  Approximation: 0.039944029379
N: 81  Approximation: 0.039983971543
N: 121  Approximation: 0.039984006406
```

```
N: 161  Approximation: 0.039984006397
```

Problem 4

Use iterated inverse interpolation to find an approximation to the solution of x - e^(-x) = 0 using the data:

```
%       x   | e^(-x)
%       -------------------
%       0.3 | 0.740818
%       0.4 | 0.670320
%       0.5 | 0.606531
%       0.6 | 0.548812
%       0.7 | 0.496585
fprintf('Problem 4\n');

y = 0;
x = [0.3:.1:.7]; % start at 0.3, increment by 0.1, end at 0.7
eToTheMinusX = [0.740818, 0.670320, 0.606531, 0.548812, 0.496585];
yk = x - eToTheMinusX; % outputs for f(x) = x - e^(-x) so input for
 f-1(x)
n = length(x);

Qij = NevillesMethod(y, yk, x);
approximation = Qij(n,n);
fprintf('The approximate value of x such that x - e^(-x) = 0 is x =
 %.6f\n\n',approximation)
```

```
Problem 4
The approximate value of x such that x - e^(-x) = 0 is x = 0.567144
```

Get Answer for Problem 1 Function

```
function approximation = GetAnswerP1(x, indexingArray)
% This function runs Nevilles method and prints results using the data
% table for problem 1.
%    INPUTS- x: input for function f
%            indexingArray: array used to determine the order of the
 inVal
%            and outVal vectors. Will be user generated and depend on
 value
%            of x.
%   OUTPUTS- approximation: approximate solution based on
 indexingArray and
%            results of Neville's method.

% x (inVals) and f(x) (outVals) as given in the table in pdf
inVals = [0.5; 0.7; 1.0; 1.3; 1.5; 1.6; 2.0];
outVals = [1.772454; 1.298055; 1.000000; 0.897471; 0.886227;
 0.893515,;1.000000];

% order of values depending on indexingArray provided by user
```

```
inVals = inVals(indexingArray);
outVals = outVals(indexingArray);

% Run Neville's Method and obtain the table of outputs.
Qij = NevillesMethod(x, inVals, outVals); % gets table of
 approximations

% Final approximation will appear in the lower right-hand corner
approximation = Qij(7,7);

end
```

Get Answer for Problem 2 and 3 Function

```
function approximation = GetAnswerP2and3(n, x, pnum)
% This function generates nodes in [-5,5] then
% finds the corresponding outputs from the function:
%       f(x) = 1 / ( 1 + (x^2) )
% It then approximates x using that set of data. NOTE: the data WILL
 NOT be
% reindexed for these runs; it will be AS GENERATED by linspace and
 f(x)
%    INPUTS- n: number of evenly spaced nodes
%           x: value for which to find approximation
%           pnum: problem number determines function and node spacing.
%                   2: generates nodes evenly spaced nodes
%                   3: generates Chebyshev nodes

%   OUTPUTS- approximation: approximation based on n evenly spaced
 nodes
%   and f(x) using Neville's Method.

f = @(x) 1 ./ ( 1 + (x.^2) );
if pnum == 2
    xVals = linspace(-5,5, n);
elseif pnum ==3
    chebyshevNodes = @(k) -5 * cos( ( (2*k-1) / (2*n) )*pi );
    xVals = [1:n];
    for m = 1: n
        xVals(m) = chebyshevNodes(m);
    end
end


yVals = f(xVals);

Qij = NevillesMethod(x, xVals, yVals);
% Neville's Method will generate an n x n matrix where n is the number
 of
% data points so the lower-right corner will always be Qij(n,n)
approximation = Qij(n,n);

end
```

Neville's Method Function

```
function Qij = NevillesMethod(x, inVals, outVals)
% Function to run Neville's Iterated Interpolation Method
%    INPUTS- x: value for which to find approximation
%            inVals: x1, x2, ... , xn
%            outVals: f(x1), f(x2), ... , f(xn)
%   OUTPUTS- Qij: matrix where Qij(n,n) is approximate solution

% verify that inVals and outVals are the same length. It won't work
% otherwise.
if length(inVals) ~= length(outVals)
    fprintf('The size of your inVals and outVals vectors needs to be
 the same\n');
    Qij = NaN;
    return
end

numOfDataPoints = length(outVals);
%Initialize matrix with zeros
Qij = zeros(numOfDataPoints,numOfDataPoints);
% Fill first column with f(x) values
for i = 1: numOfDataPoints
    Qij(i,1) = outVals(i);
end
%Start Neville's method
for i = 2: numOfDataPoints
    xi = inVals(i);
    for j = 1: i-1
        xij = inVals(i-j);
        nextVal = ((x - xij).*Qij(i,j) - (x - xi).*Qij(i-1, j)) / (xi
 - xij);
        Qij(i,j+1) =  nextVal;
        %{
        fprintf('((%.2f - %.8f)*(%.8f) - (%.1f - %.8f)*(%.8f))/(%.8f -
 %.8f)\n', x, xij, Qij(i-1, j), x, xi,Qij(i,j),xi,xij)
        Qij
        %}
    end
end
end
```

```
The approximate value of f(0.8) = 1.163081
The approximate value of f(1.2) = 0.918424
The approximate value of f(1.7) = 0.908099
```

Published with MATLAB® R2018b