Table of Contents

Part 1: Neville's Method

Approximates values using Neville's Method and provided data

```
fprintf('Part 1\n');

% Initial Data
x = [ 0.5 ; 0.7 ; 1.0 ; 1.3 ; 1.5 ; 1.6 ; 2.0 ];
data = [ 1.772454 ; 1.298055 ; 1.000000 ; 0.897471 ; 0.886227 ;
 0.893515 ; 1.000000 ];

Zeros = zeros(size(data, 1), size(data, 1)-1);

% Combines initial data with room for future Q values
Q = [data Zeros];

% Value to approximate
value = 0.8;

for i = 1:size(Q, 1)-1 % -1 is because n = the number of points minus
 1
    for j = 1:i
        % Because Matlab starts at an index of 1, all indexs had 1
 added to them to compensate
        Q(i+1, j+1) = (((value-x(i-j+1))*Q(i+1, j)) - ((value-x(i
+1))*Q(i, j)))/ (x(i+1)-x(i-j+1));
    end
end
answer = Q(size(data, 1), size(data, 1));
fprintf('Approximation for f(0.8): %.6f\n', answer);

% Value to approximate
value = 1.2;

for i = 1:size(Q, 1)-1 % -1 is because n = the number of points minus
 1
    for j = 1:i
        % Because Matlab starts at an index of 1, all indexs had 1
 added to them to compensate
        Q(i+1, j+1) = (((value-x(i-j+1))*Q(i+1, j)) - ((value-x(i
+1))*Q(i, j)))/ (x(i+1)-x(i-j+1));
    end
end
answer = Q(size(data, 1), size(data, 1));
```

```
fprintf('Approximation for f(1.2): %.6f\n', answer);

% Value to approximate
value = 1.7;

for i = 1:size(Q, 1)-1 % -1 is because n = the number of points minus
 1
    for j = 1:i
        % Because Matlab starts at an index of 1, all indexs had 1
 added to them to compensate
        Q(i+1, j+1) = (((value-x(i-j+1))*Q(i+1, j)) - ((value-x(i
+1))*Q(i, j)))/ (x(i+1)-x(i-j+1));
    end
end
answer = Q(size(data, 1), size(data, 1));
fprintf('Approximation for f(1.7): %.6f\n', answer);
```

```
Part 1
Approximation for f(0.8): 1.163081
Approximation for f(1.2): 0.918424
Approximation for f(1.7): 0.908099
```

Part 2: Generate Function Data and Use Neville's Method

Approximates values using Neville's Method and function data generate from equation

```
fprintf('Part 2\n');

N = 11;

x = linspace(-5, 5, N);
x = x';
data = (1+x.^2).^-1;
Zeros = zeros(size(data, 1), size(data, 1)-1);
% Combines initial data with room for future Q values
Q = [data Zeros];

for i = 1:size(Q, 1)-1 % -1 is because n = the number of points minus
 1
    for j = 1:i
        % Because Matlab starts at an index of 1, all indexs had 1
 added to them to compensate
        Q(i+1, j+1) = (((4.9-x(i-j+1))*Q(i+1, j)) - ((4.9-x(i+1))*Q(i,
 j))) / (x(i+1)-x(i-j+1));
    end
end
answer = Q(size(Q, 1), size(Q, 1));
fprintf('N = %d, Approximation: %.8f\n', N, answer);

N = 21;
```

```
x = linspace(-5, 5, N);
x = x';
data = 1./(1+x.^2);
Zeros = zeros(size(data, 1), size(data, 1)-1);
% Combines initial data with room for future Q values
Q = [data Zeros];

for i = 1:size(Q, 1)-1 % -1 is because n = the number of points minus
 1
    for j = 1:i
        % Because Matlab starts at an index of 1, all indexs had 1
 added to them to compensate
        Q(i+1, j+1) = (((4.9-x(i-j+1))*Q(i+1, j)) - ((4.9-x(i+1))*Q(i,
 j)))/ (x(i+1)-x(i-j+1));
    end
end
answer = Q(size(Q, 1), size(Q, 1));
fprintf('N = %d, Approximation: %.8f\n', N, answer);

N = 41;

x = linspace(-5, 5, N);
x = x';
data = 1./(1+x.^2);
Zeros = zeros(size(data, 1), size(data, 1)-1);
% Combines initial data with room for future Q values
Q = [data Zeros];

for i = 1:size(Q, 1)-1 % -1 is because n = the number of points minus
 1
    for j = 1:i
        % Because Matlab starts at an index of 1, all indexs had 1
 added to them to compensate
        Q(i+1, j+1) = (((4.9-x(i-j+1))*Q(i+1, j)) - ((4.9-x(i+1))*Q(i,
 j)))/ (x(i+1)-x(i-j+1));
    end
end
answer = Q(size(Q, 1), size(Q, 1));
fprintf('N = %d, Approximation: %.8f\n', N, answer);

N = 81;

x = linspace(-5, 5, N);
x = x';
data = 1./(1+x.^2);
Zeros = zeros(size(data, 1), size(data, 1)-1);
% Combines initial data with room for future Q values
Q = [data Zeros];

for i = 1:size(Q, 1)-1 % -1 is because n = the number of points minus
 1
    for j = 1:i
        % Because Matlab starts at an index of 1, all indexs had 1
 added to them to compensate
```

```
        Q(i+1, j+1) = (((4.9-x(i-j+1))*Q(i+1, j)) - ((4.9-x(i+1))*Q(i,
 j)))/ (x(i+1)-x(i-j+1));
    end
end
answer = Q(size(Q, 1), size(Q, 1));
fprintf('N = %d, Approximation: %.8f\n', N, answer);

N = 121;

x = linspace(-5, 5, N);
x = x';
data = 1./(1+x.^2);
Zeros = zeros(size(data, 1), size(data, 1)-1);
% Combines initial data with room for future Q values
Q = [data Zeros];

for i = 1:size(Q, 1)-1 % -1 is because n = the number of points minus
 1
    for j = 1:i
        % Because Matlab starts at an index of 1, all indexs had 1
 added to them to compensate
        Q(i+1, j+1) = (((4.9-x(i-j+1))*Q(i+1, j)) - ((4.9-x(i+1))*Q(i,
 j)))/ (x(i+1)-x(i-j+1));
    end
end
answer = Q(size(Q, 1), size(Q, 1));
fprintf('N = %d, Approximation: %.8f\n', N, answer);
```

```
Part 2
N = 11, Approximation: 1.23031656
N = 21, Approximation: -58.23814110
N = 41, Approximation: -78688.99750112
N = 81, Approximation: -40443044569.41036224
N = 121, Approximation: 35481700987184308.00000000
```

Part 3: Chebyshev Nodes

Approximate values using Chebyshev Nodes

```
fprintf('Part 3\n');

N = 11;

x = linspace(-5, 5, N);
x = x';
data = -5*cos((((2.*x)-1)/(2.*N))*pi);
Zeros = zeros(size(data, 1), size(data, 1)-1);
% Combines initial data with room for future Q values
Q = [data Zeros];

for i = 1:size(Q, 1)-1 % -1 is because n = the number of points minus
 1
    for j = 1:i
```

```
        % Because Matlab starts at an index of 1, all indexs had 1
 added to them to compensate
        Q(i+1, j+1) = (((4.9-x(i-j+1))*Q(i+1, j)) - ((4.9-x(i+1))*Q(i,
 j))) / (x(i+1)-x(i-j+1));
    end
end
answer = Q(size(Q, 1), size(Q, 1));
fprintf('N = %d, Approximation: %.8f\n', N, answer);

N = 21;

x = linspace(-5, 5, N);
x = x';
data = -5*cos((((2.*x)-1)/(2.*N))*pi);
Zeros = zeros(size(data, 1), size(data, 1)-1);
% Combines initial data with room for future Q values
Q = [data Zeros];

for i = 1:size(Q, 1)-1 % -1 is because n = the number of points minus
 1
    for j = 1:i
        % Because Matlab starts at an index of 1, all indexs had 1
 added to them to compensate
        Q(i+1, j+1) = (((4.9-x(i-j+1))*Q(i+1, j)) - ((4.9-x(i+1))*Q(i,
 j)))/ (x(i+1)-x(i-j+1));
    end
end
answer = Q(size(Q, 1), size(Q, 1));
fprintf('N = %d, Approximation: %.8f\n', N, answer);

N = 41;

x = linspace(-5, 5, N);
x = x';
data = -5*cos((((2.*x)-1)/(2.*N))*pi);
Zeros = zeros(size(data, 1), size(data, 1)-1);
% Combines initial data with room for future Q values
Q = [data Zeros];

for i = 1:size(Q, 1)-1 % -1 is because n = the number of points minus
 1
    for j = 1:i
        % Because Matlab starts at an index of 1, all indexs had 1
 added to them to compensate
        Q(i+1, j+1) = (((4.9-x(i-j+1))*Q(i+1, j)) - ((4.9-x(i+1))*Q(i,
 j)))/ (x(i+1)-x(i-j+1));
    end
end
answer = Q(size(Q, 1), size(Q, 1));
fprintf('N = %d, Approximation: %.8f\n', N, answer);

N = 81;

x = linspace(-5, 5, N);
```

```
x = x';
data = -5*cos((((2.*x)-1)/(2.*N))*pi);
Zeros = zeros(size(data, 1), size(data, 1)-1);
% Combines initial data with room for future Q values
Q = [data Zeros];

for i = 1:size(Q, 1)-1 % -1 is because n = the number of points minus
 1
    for j = 1:i
        % Because Matlab starts at an index of 1, all indexs had 1
 added to them to compensate
        Q(i+1, j+1) = (((4.9-x(i-j+1))*Q(i+1, j)) - ((4.9-x(i+1))*Q(i,
 j)))/ (x(i+1)-x(i-j+1));
    end
end
answer = Q(size(Q, 1), size(Q, 1));
fprintf('N = %d, Approximation: %.8f\n', N, answer);

N = 121;

x = linspace(-5, 5, N);
x = x';
data = -5*cos((((2.*x)-1)/(2.*N))*pi);
Zeros = zeros(size(data, 1), size(data, 1)-1);
% Combines initial data with room for future Q values
Q = [data Zeros];

for i = 1:size(Q, 1)-1 % -1 is because n = the number of points minus
 1
    for j = 1:i
        % Because Matlab starts at an index of 1, all indexs had 1
 added to them to compensate
        Q(i+1, j+1) = (((4.9-x(i-j+1))*Q(i+1, j)) - ((4.9-x(i+1))*Q(i,
 j)))/ (x(i+1)-x(i-j+1));
    end
end
answer = Q(size(Q, 1), size(Q, 1));
fprintf('N = %d, Approximation: %.8f\n', N, answer);
```

```
Part 3
N = 11, Approximation: -1.54508497
N = 21, Approximation: -3.95535517
N = 41, Approximation: -4.71851254
N = 81, Approximation: 26391.22194366
N = 121, Approximation: 1274093872581498.00000000
```

Part 4: Inverse Interpolation

Approximate the solution using iterated inverse interpolation and provided data

```
fprintf('Part 4\n');

% equation is y = x-e^-x
x = [0.3;      0.4;      0.5;      0.6;      0.7];
```

```
e = [0.740818; 0.670320; 0.606531; 0.548812; 0.496585]; % e is e^-x
y = x-e;

data = x;
Zeros = zeros(size(data, 1), size(data, 1)-1);
% Combines initial data with room for future Q values
Q = [data Zeros];

for i = 1:size(Q, 1)-1 % -1 is because n = the number of points minus
 1
    for j = 1:i
        % Because Matlab starts at an index of 1, all indexs had 1
 added to them to compensate
        Q(i+1, j+1) = (((0-y(i-j+1))*Q(i+1, j)) - ((0-y(i+1))*Q(i,
 j))) / (y(i+1)-y(i-j+1));
    end
end
answer = Q(size(Q, 1), size(Q, 1));
fprintf('Approximation: %.6f\n', answer);
```

```
Part 4
Approximation: 0.567144
```

Published with MATLAB® R2020a