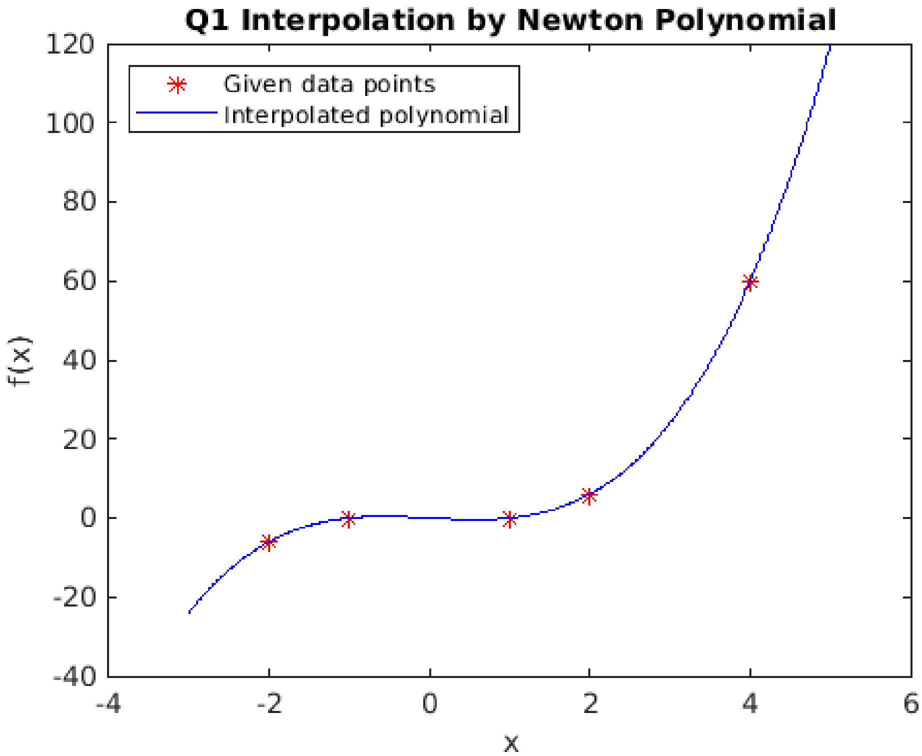
Table of Contents

Q1: Newton polynomial of (-2, -6), (-1,0), (1, 0), (2, 6), (4, 60)

```
% points
x1 = [-2 -1 1 2 4];
y1 = [-6 0 0 6 60];

% Calculate divided diff. table & newton poly coeff.
divDifTab = divDiff(x1, y1);
newPolCoef = funcNewPoly(divDifTab, x1);

% Printing values and plotting
disp('Divided Difference table : ')
disp(divDifTab)
disp('Newton Polynomial oefficients : ')
disp(newPolCoef)

xValues = linspace(-3, 5, 500);
yValues = polyval(newPolCoef, xValues);

figure(1);

plot(x1, y1, 'r*', xValues, yValues, 'b-')
title('Q1 Interpolation by Newton Polynomial')
legend('Given data points', 'Interpolated
 polynomial', 'Location', 'northwest')
xlabel('x')
ylabel('f(x)')
```

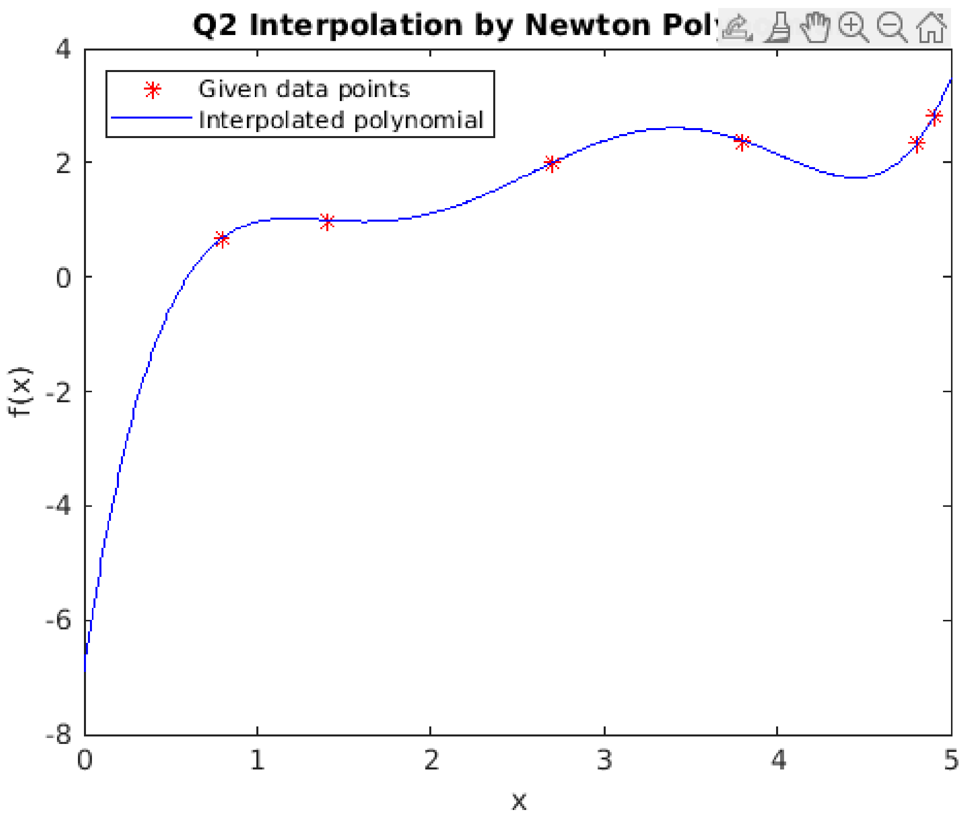


Q2: Newton polynomial of (0.8, 0.69), (1.4, 1.0), (2.7, 2.0), (3.8, 2.39), (4.8, 2.34), (4.9, 2.83)

```
% points
x2 = [0.8 1.4 2.7 3.8 4.8 4.9];
y2 = [0.69 1.00 2.00 2.39 2.34 2.83];

% Calculating the Divided Difference Table and Newton polynomial
 coefficients
divDifTab = divDiff(x2, y2);
newPolCoef = funcNewPoly(divDifTab, x2);

% Printing values and plotting
disp('Divided Difference table : ')
disp(divDifTab)
disp('Newton Polynomial coefficients : ')
disp(newPolCoef)

xValues = linspace(0, 5, 500);
yValues = polyval(newPolCoef, xValues);

figure(2);

plot(x2, y2, 'r*', xValues, yValues, 'b-')
title('Q2 Interpolation by Newton Polynomial')
```

```
legend('Given data points', 'Interpolated
 polynomial', 'Location', 'northwest')
xlabel('x')
ylabel('f(x)')
```

```
Divided Difference table :
    0.6900    0.5167    0.1329   -0.1019    0.0240    0.1432
    1.0000    0.7692   -0.1728   -0.0058    0.6111         0
    2.0000    0.3545   -0.1926    2.1330         0         0
    2.3900   -0.0500    4.5000         0         0         0
    2.3400    4.9000         0         0         0         0
    2.8300         0         0         0         0         0

Newton Polynomial coefficients :
    0.1432   -1.9091    9.3460  -20.6759   20.9512   -6.8885
```

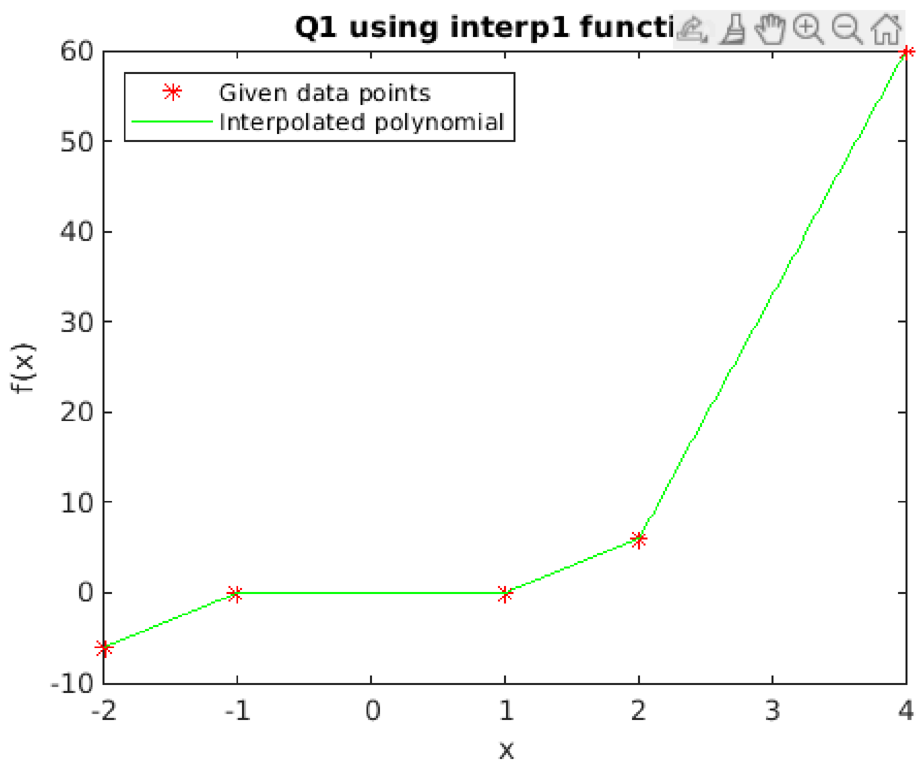


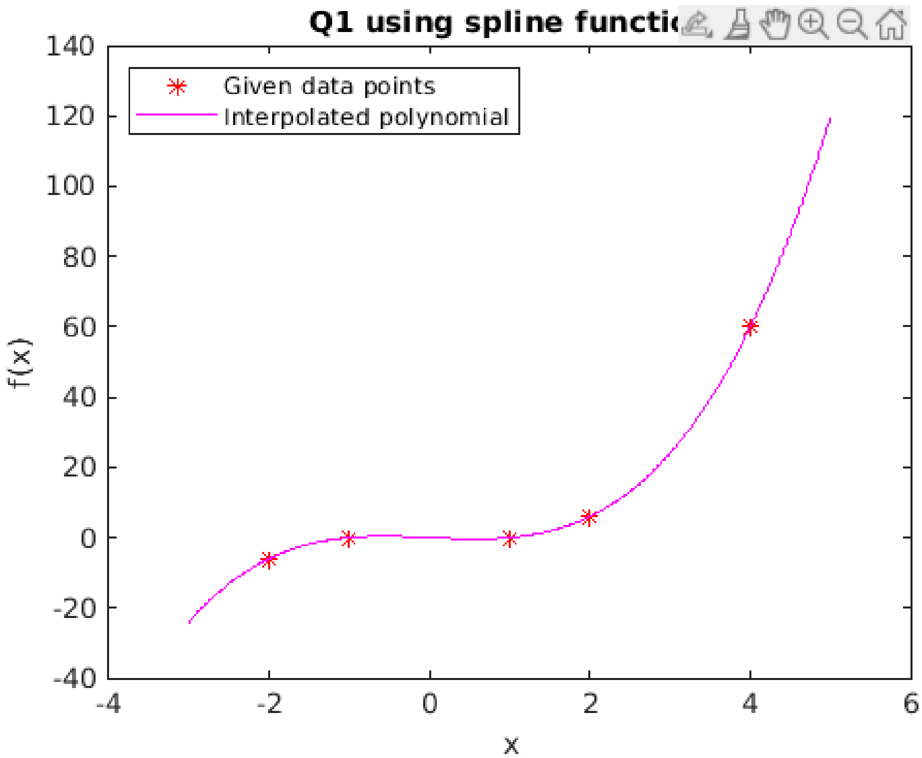
Q3: Exploring in-built interpolation techniques

Q1 using in-built interpolation techniques

```
% Q1-interp1
figure(3);

xValues = linspace(-3, 5, 500);
```

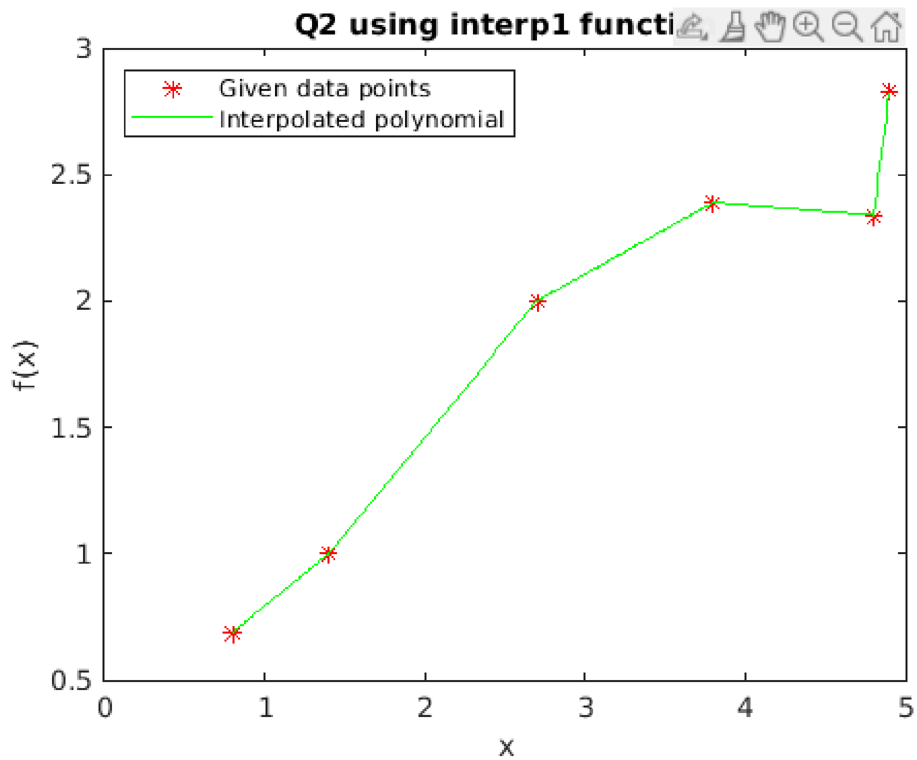
```
yValues = interp1(x1, y1, xValues);
plot(x1, y1, 'r*', xValues, yValues, 'g-');
title('Q1 using interp1 function')
legend('Given data points', 'Interpolated
 polynomial', 'Location', 'northwest')
xlabel('x')
ylabel('f(x)')

% Q1-spline
figure(5);

xValues = linspace(-3, 5, 500);
yValues = spline(x1, y1, xValues);
plot(x1, y1, 'r*', xValues, yValues, 'm-');
title('Q1 using spline function')
legend('Given data points', 'Interpolated
 polynomial', 'Location', 'northwest')
xlabel('x')
ylabel('f(x)')
```

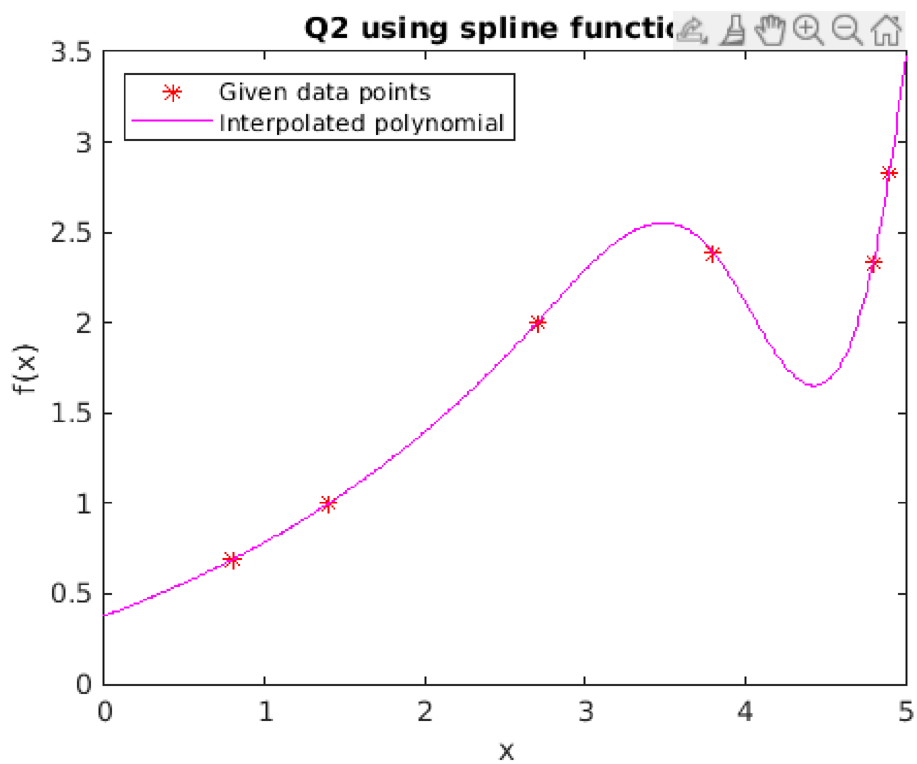




Q2 using in-built interpolation techniques

```
% Q2-interp1
figure(4);

xValues = linspace(0, 5, 500);
yValues = interp1(x2, y2, xValues);
plot(x2, y2, 'r*', xValues, yValues, 'g-');
title('Q2 using interp1 function')
legend('Given data points', 'Interpolated
 polynomial', 'Location', 'northwest')
xlabel('x')
ylabel('f(x)')

% Q2-spline

figure(6);

xValues = linspace(0, 5, 500);
yValues = spline(x2, y2, xValues);
plot(x2, y2, 'r*', xValues, yValues, 'm-');
title('Q2 using spline function')
legend('Given data points', 'Interpolated
 polynomial', 'Location', 'northwest')
xlabel('x')
ylabel('f(x)')
```


Q2 using interp1 functi
Given data points
Interpolated polynomial
f(x)
x
0
1
2
3
4
5
0.5
1
1.5
2
2.5
3


Q2 using spline functi
Given data points
Interpolated polynomial
f(x)
x
0
1
2
3
4
5
0
0.5
1
1.5
2
2.5
3
3.5

Function definitions

Divided Difference table funciton

calculates the divided difference table

```
function divDifTab = divDiff(x, y)
    n = length(x) - 1;
    divDifTab = zeros(n + 1, n + 1);
    divDifTab(1 : n + 1, 1) = y';
    for i = 2 : n + 1
        for j = 1 : n - i + 2
            divDifTab(j, i) = (divDifTab(j+1, i-1) - divDifTab(j,
 i-1))/(x(i+j - 1) - x(j));    % returns a matrix of the divided
 differences.
        end
    end
end
```

Newton's Polynomial Coefficients funciton

Calcultes Newton Polynomial coefficients

```
function newPolCoef = funcNewPoly(divDifTab, x)
    n = length(x) - 1;
    a = divDifTab(1, :);
    newPolCoef = a(n+1);
    for i = n:-1:1
        newPolCoef = [newPolCoef a(i)] - [0 newPolCoef*x(i)];   %
 Returns a vector of polynomial coefficients
    end
end
```

```
Divided Difference table :
    -6     6    -2     1     0
     0     0     2     1     0
     0     6     7     0     0
     6    27     0     0     0
    60     0     0     0     0

Newton Polynomial oefficients :
     0     1     0    -1     0
```

Published with MATLAB® R2021a