Table of Contents

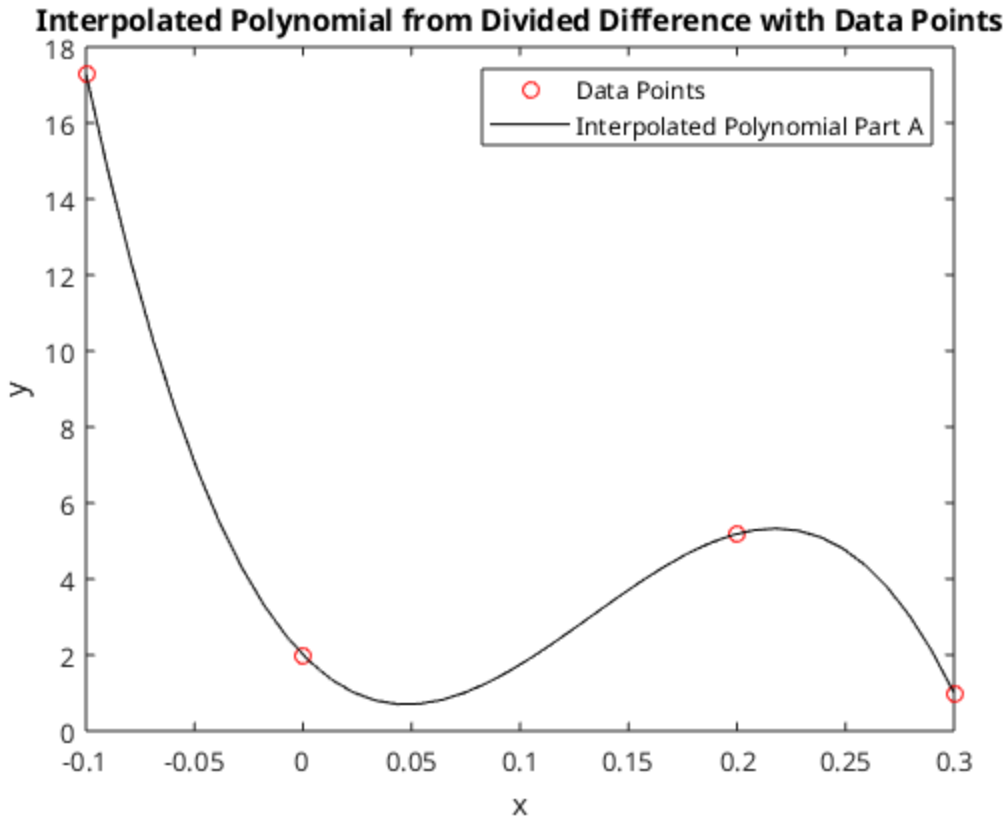
Part 1: Estimation using divided differences

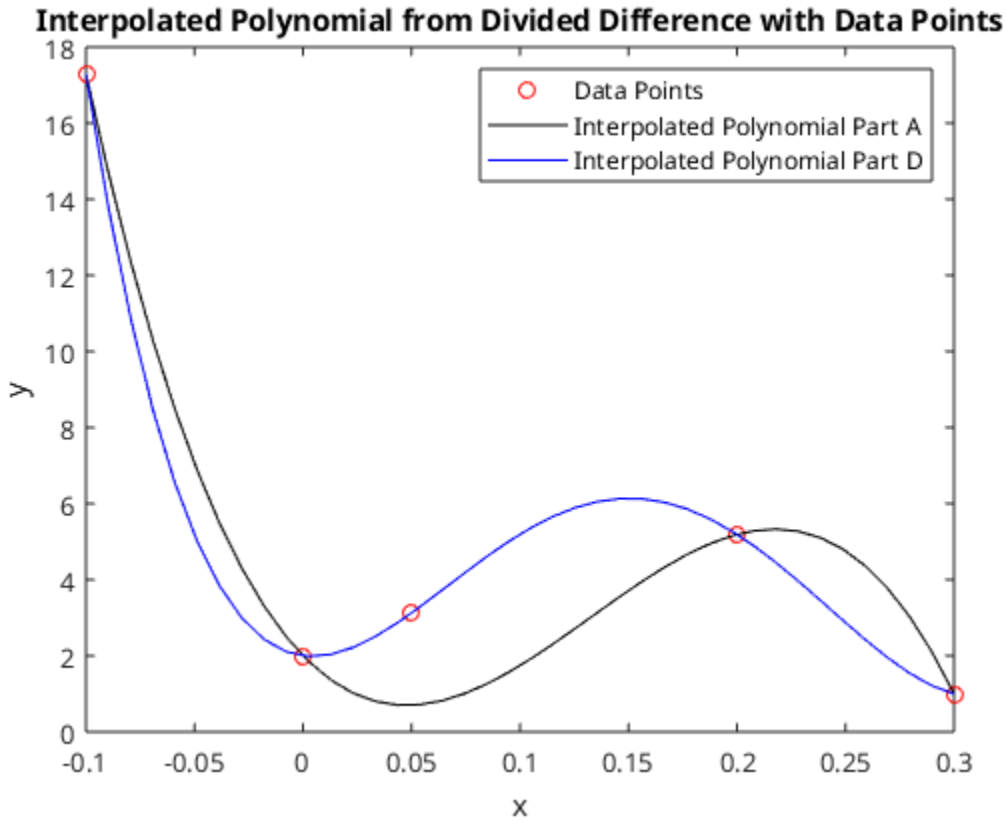
Uses divided differences to calculate an approximation and graph it

```
fprintf("Part 1: Estimation using divided differences\n");

x = [-0.10; 0.00; 0.20; 0.30];
fx = [17.3000; 2.0000; 5.1900; 1.0000];
Zeros = zeros(size(fx, 1), size(fx, 1)-1);
F = [fx Zeros];

for i = 1:size(x)-1 %Time run is n which since x starts at x0 is size
+1
    for j = 1:i
        F(i+1,j+1) = (F(i+1,j)-F(i,j))./(x(i+1)-x(i-j+1));
    end
end
coefficients = diag(F);

P = @(y) coefficients(1) + dot(coefficients(2:end), [prod(y-x(1:1)),
 prod(y-x(1:2)), prod(y-x(1:3))]);
fprintf("Approximation of 0.1: %.6f\n", P(.1));
fprintf("Approximation of 0.4: %.6f\n", P(.4));

w = linspace(-.1,.3,40)';
Pw = zeros(size(w));
for i=1:size(w)
    Pw(i) = P(w(i));
end

figure;
plot(x, fx, 'or');
hold on;
plot(w, Pw, '-k');
hold off;
xlabel('x');
ylabel('y');
title('Interpolated Polynomial from Divided Difference with Data
 Points');
legend('Data Points', 'Interpolated Polynomial Part A');

x = [-0.10; 0.00; 0.05; 0.20; 0.30];
fx = [17.3000; 2.0000; 3.125; 5.1900; 1.0000];
Zeros = zeros(size(fx, 1), size(fx, 1)-1);
```

```
F = [fx Zeros];

for i = 1:size(x)-1 %Time run is n which since x starts at x0 is size
+1
    for j = 1:i
        F(i+1,j+1) = (F(i+1,j)-F(i,j))./(x(i+1)-x(i-j+1));
    end
end
coefficients = diag(F);

P = @(y) coefficients(1) + dot(coefficients(2:end), [prod(y-x(1:1)),
 prod(y-x(1:2)), prod(y-x(1:3)), prod(y-x(1:4))]);
fprintf("Approximation of 0.1 with 5 points: %.6f\n", P(.1));
fprintf("Approximation of 0.4 with 5 points: %.6f\n", P(.4));

Pw2 = zeros(size(w));
for i=1:size(w)
    Pw2(i) = P(w(i));
end

figure;
plot(x, fx, 'or');
hold on;
plot(w, Pw, '-k');
plot(w, Pw2, '-b');
hold off;
xlabel('x');
ylabel('y');
title('Interpolated Polynomial from Divided Difference with Data
 Points');
legend('Data Points', 'Interpolated Polynomial Part A', 'Interpolated
 Polynomial Part D');
```

```
Part 1: Estimation using divided differences
Approximation of 0.1: 1.743333
Approximation of 0.4: -22.166667
Approximation of 0.1 with 5 points: 5.192222
Approximation of 0.4 with 5 points: 12.322222
```


Interpolated Polynomial from Divided Difference with Data Points
Data Points
Interpolated Polynomial Part A
y
x
18
16
14
12
10
8
6
4
2
0
-0.1
-0.05
0
0.05
0.1
0.15
0.2
0.25
0.3


Interpolated Polynomial from Divided Difference with Data Points
Data Points
Interpolated Polynomial Part A
Interpolated Polynomial Part D
y
x
18
16
14
12
10
8
6
4
2
0
-0.1
-0.05
0
0.05
0.1
0.15
0.2
0.25
0.3

Part 2: Kentucky Derby Estimation

Predicts the time at the 3/4 mile pole

```
fprintf("Part 2: Kentucky Derby Estimation\n");

x = [.25; .5; 1; 1.25];
fx = [25.2; 49.2; 96.4; 119.4];
Zeros = zeros(size(fx, 1), size(fx, 1)-1);
F = [fx Zeros];

for i = 1:size(x)-1 %Time run is n which since x starts at x0 is size
+1
    for j = 1:i
        F(i+1,j+1) = (F(i+1,j)-F(i,j))./(x(i+1)-x(i-j+1));
    end
end
coefficients = diag(F);

P = @(y) coefficients(1) + dot(coefficients(2:end), [prod(y-x(1:1)),
 prod(y-x(1:2)), prod(y-x(1:3))]);
answer = P(.75);
fprintf("Approximation of time (in seconds) at 3/4 mile mark: %.2f\n",
 answer);
fprintf("The actual time was 73 seconds. This means the relative error
 was %.2f%%\n", 100*abs((answer-73)/73));
```

```
Part 2: Kentucky Derby Estimation
Approximation of time (in seconds) at 3/4 mile mark: 72.97
The actual time was 73 seconds. This means the relative error was
 0.05%
```

Part 3: Degree of Polynomial

Uses divided differences to find the degree of the polynomial to interpolate a function

```
fprintf("Part 3: Degree of Polynomial\n");

x = [0; 1; 2; 3; 4; 5; 6; 7];
fx = [0; -2; -8; 0; 64; 250; 648; 1372];
Zeros = zeros(size(fx, 1), size(fx, 1)-1);
F = [fx Zeros];

for i = 1:size(x)-1 %Time run is n which since x starts at x0 is size
+1
    for j = 1:i
        F(i+1,j+1) = (F(i+1,j)-F(i,j))./(x(i+1)-x(i-j+1));
    end
end
coefficients = diag(F);
syms value;
simplify(coefficients(1) + dot(coefficients(2:end), [(value-x(1)),
 (value-x(1))*(value-x(2)), (value-x(1))*(value-x(2))*(value-x(3)),
```

```
 (value-x(1))*(value-x(2))*(value-x(3))*(value-x(4)), (value-
x(1))*(value-x(2))*(value-x(3))*(value-x(4))*(value-x(5)), (value-
x(1))*(value-x(2))*(value-x(3))*(value-x(4))*(value-x(5))*(value-
x(6)), (value-x(1))*(value-x(2))*(value-x(3))*(value-x(4))*(value-
x(5))*(value-x(6))*(value-x(6))]))
fprintf("The equation is x^3*(x-3) which equals x^4-3x^3\n");
fprintf("Therefore the degree of the polynomial is 4");
```

```
Part 3: Degree of Polynomial

ans =

value^3*(value - 3)

The equation is x^3*(x-3) which equals x^4-3x^3
Therefore the degree of the polynomial is 4
```

Published with MATLAB® R2020a