Numerical Methods

**Report**

Date of the exercise: **28/03/2019**

Exercise: **Interpolation**

Group: 2, Team:

Subsection (names):

1. Mateusz Nowotnik

2. Dawid Tomala

Solving the interpolation problem of calculating the interpolation function value using the Lagrange and Chebyshev method.

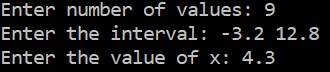
1. #undef \_\_STRICT\_ANSI\_\_
2. #include <cmath>
3. #include <windows.h>
4. #include <iostream>
5. **using** **namespace** std;
7. **double** f(**double** x)
8. {
9. **return** 5/(x\*x+2);
10. }
12. **double** F(**double** xi[], **double** yi[], **double** wi[], **int** n, **double** x, **bool** finalWi)
13. {
14. **double** F = 0;
15. **for**(**int** i = 0; i <= n; i ++ )
16. {
17. wi[i] = 1;
18. **for**(**int** j = 0; j<=n;j++)
19. {
20. **if**(j!=i)
21. // the auxiliary polynomials
22. wi[i] \*= (x-xi[j])/(xi[i]-xi[j]);
23. }
24. F += wi[i]\*yi[i];
25. }
27. **if**(finalWi == **true**) {
28. **for** (**int** i = 0; i < n+1; i++) {
29. cout << "wi[" << i << "] : "<< wi[i] << endl;
30. }
31. }
33. **return** F;
34. }
36. **void** maxError(**double** xi[], **double** yi[], **int** a, **int** b, **int** n, **double** wi[], **double** x, **bool** finalWi)
37. {
38. // find max error in the interval determined by a and b with step equal to 0.01
39. **double** err1;
40. **double** i = a;
41. **double** err2 = f(a) - F(xi, yi, wi, n, x, finalWi);
42. **while**(i <= b) {
43. i+=0.01;
44. err1 = f(a) - F(xi, yi, wi, n, x, finalWi);
45. err2 = max(err2, err1);
46. }
48. cout << "Max error: " << fabs(err2) << endl;
49. }
51. **int** main()
52. {
53. **int** n;
54. **double** a, b, x, arg;
55. **double** inter = 0;
56. **bool** finalWi = **false**;
58. cout << "Enter number of values: ";
59. cin >> n;
61. cout << "Enter the interval: ";
62. cin >> a; cin >> b;
64. cout << "Enter the value of x: ";
65. cin >> x;
67. **if** ((x < a) || (x > b)) {
68. cout << "\nFAIL: x is outside of the interval!\n";
69. **return** **false**;
70. }
72. **double** xiLag[n+1];
73. **double** xiCheb[n+1];
74. **double** yiLag[n+1];
75. **double** yiCheb[n+1];
76. **double** wi[n+1];
77. **double** ri[n+1];
79. // step, xi[0] xi[n+1] and their function values for the Lagrange method
80. inter = (b-a)/(n);
81. xiLag[0] = a;
82. xiLag[n+1] = b;
83. yiLag[0] = f(xiLag[0]);
84. yiLag[n+1] = f(xiLag[n+1]);
86. xiCheb[0] = a;
87. xiCheb[n+1] = b;
88. yiCheb[0] = f(xiCheb[0]);
89. yiCheb[n+1] = f(xiCheb[n+1]);
90. arg = M\_PI/(2\*n+2);
92. **for** (**int** i = 1; i <= n+1; i++) {
93. // Lagrange xi, yi
94. xiLag[i] = xiLag[i-1] + inter;
95. yiLag[i] = f(xiLag[i]);
97. // Chebyshev xi, yi
98. xiCheb[i] = 0.5\*((b-a)\*cos(arg\*(2\*n+1-2\*i))+b+a);
99. yiCheb[i] = f(xiCheb[i]);
100. }
102. finalWi = **true**;
104. // Task 1
105. cout << "\nLagrange: " << endl;
106. cout << "\nInterpolation function: " << F(xiLag, yiLag, wi, n, x, finalWi) << endl;
107. cout << "Interpolated function: " << f(x) << endl << endl;
109. // Task 2
110. cout << "Chebyshev: " << endl;
111. cout << "\nInterpolation function: " << F(xiCheb, yiCheb, wi, n, x, finalWi) << endl;
112. cout << "Interpolated function: " << f(x) << endl << endl;
114. finalWi = **false**;
115. // Task 3
116. maxError(xiCheb, yiCheb, a, b, n, wi, x, finalWi);
118. system("PAUSE");
120. **return** 0;
121. }

First part of the program calculates the Lagrange’s auxiliary polynomials and the **function values**  of the arguments , and from that, the **interpolation function value**; having as the inputs:

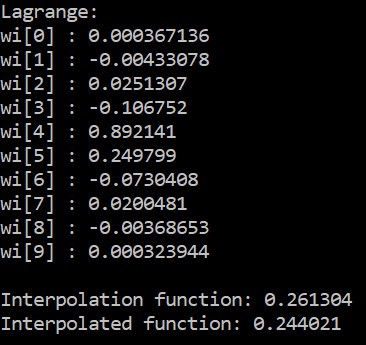
* the number of nodes
* interval [a,b]
* the value x

***Example:***

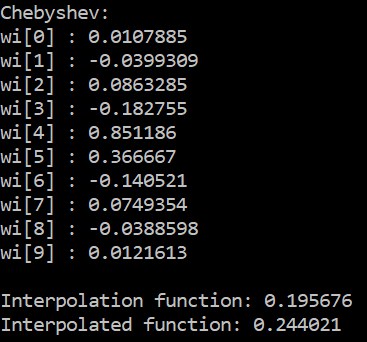
Giving to the program following inputs:



The first part of the program yields:



In the second task, for the same inputs given by the user, the program calculates roots of the Chebyshev polynomial and Chebyshev (interpolation) nodes, as well as the values of the interpolation function at the points where those nodes exist. The output is:



And for the third task, the **maximum error** is being found, subtracting the value of the interpolation function from the value of the interpolated function for the argument in the interval determined by and with the step equal to ,which gives the greatest absolute value of that subtraction.

Output:

