

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧЕРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Санкт-Петербургский политехнический университет Петра Великого»

ИНСТИТУТ КОМПЬЮТЕРНЫХ НАУК И ТЕХНОЛОГИЙ
Высшая школа программной инженерии

Отчет по лабораторной работе по дисциплине «Вычислительная математика»

Выполнила студентка гр. 3530904/80001

Прохорова А. И.

Руководитель

Устинов С. М.

Санкт-Петербург
2019

1 Задание

Для таблично заданной функции $f(x)$

x	-1.0	-0.9	-0.8	-0.7	-0.6	-0.5
f(x)	0.5440	-0.4121	-0.9894	-0.6570	0.2794	0.9589

построить сплайн-функцию и использовать её для нахождения корня уравнения $f(x) = 1.8 \cdot x^2$ на промежутке $[-1, -0.5]$ методом бисекции.

2 Результаты работы программы

Original function f(x):

```
-1 ; 0.544
-0.9 ; -0.4121
-0.8 ; -0.9894
-0.7 ; -0.657
-0.6 ; 0.2794
-0.5 ; 0.9589
```

Spline coefficients:

```
spline 0: -9.4901; -9.55729; 88.4833
spline 1: -8.74706; 16.9877; 127.529
spline 2: -1.52365; 55.2465; -67.7
spline 3: 7.49465; 34.9365; -162.429
spline 4: 9.60906; -13.7923; -143.483
spline 5: 2.5461; -56.8373; -143.483
```

Function given by equation $f(x) = \text{rightPart}(x)$

```
-1 ; -1.256
-0.9 ; -1.8701
-0.8 ; -2.1414
-0.7 ; -1.539
-0.6 ; -0.3686
-0.5 ; 0.5089
```

Number of roots: 1

```
-0.55
```

3 Приложение

3.1 Код lab1.hpp

```
1 #ifndef COMPUTATIONAL_MATH_LAB1
2 #define COMPUTATIONAL_MATH_LAB1
3
4 #include <iosfwd>
5 #include <vector>
6
7 #define CUSTOM_EPSILON 1e-7
8
9 struct point_t
10 {
11     double x;
```

```

12     double y;
13 };
14
15 struct function_t
16 {
17     std::vector<double> x;
18     std::vector<double> y;
19
20     function_t();
21     function_t(size_t n);
22
23     size_t size() const;
24     double* X();
25     double* Y();
26 };
27 std::istream& operator>>(std::istream& in, function_t& function);
28 std::ostream& operator<<(std::ostream& out, const function_t& function);
29
30 struct spline_t
31 {
32     std::vector<double> b;
33     std::vector<double> c;
34     std::vector<double> d;
35
36     spline_t(size_t n);
37     double* B();
38     double* C();
39     double* D();
40     size_t size() const;
41 };
42 std::ostream& operator<<(std::ostream& out, const spline_t& factors);
43
44
45
46 #endif // COMPUTATIONAL_MATH_LAB1

```

3.2 Код lab1.cpp

```

1 #include <iostream>
2 #include <iomanip>
3 #include <algorithm>
4 #include <iterator>
5
6 #include "../cmath.h"
7 #include "lab1.hpp"
8
9 double RightPart(double x)
10 {
11     return 1.8 * x * x;
12 }
13
14 double bisection(function_t& function, spline_t& factors,
15     double leftX, double rightX, double (*rightPart)(double))
16 {
17     double middleX = -1;
18     double middleY = -4;
19     int last = 5;
20
21     while (abs(middleY) > CUSTOM_EPSILON)
22     {
23         middleX = (leftX + rightX) / 2;
24         middleY = seval(function.size(), middleX,

```

```

25     function.X(), function.Y(),
26     factors.B(), factors.C(), factors.D(), &last) -
27     rightPart(middleX);
28
29     double rightY = seval(function.size(), rightX,
30     function.X(), function.Y(),
31     factors.B(), factors.C(), factors.D(), &last) -
32     rightPart(rightX);
33     double leftY = seval(function.size(), leftX,
34     function.X(), function.Y(),
35     factors.B(), factors.C(), factors.D(), &last) -
36     rightPart(leftX);
37
38     if (leftY * rightY > 0)
39     {
40         std::cout << leftY << " " << rightY << "\n";
41         throw std::runtime_error("LeftY and RightY must be of different signs.");
42     }
43     if (leftY * middleY < 0)
44     {
45         rightX = middleX;
46     }
47     else if (middleY * rightY < 0)
48     {
49         leftX = middleX;
50     }
51     else
52     {
53         throw std::runtime_error("middleY must have different sign with one of the
border.");
54     }
55 }
56 return middleX;
57 }
58
59
60
61 int main()
62 {
63     function_t function;
64     std::cin >> function;
65     std::cout << "Original function f(x):\n" << function << "\n";
66
67     spline_t factors(function.size());
68
69     int flag = 0;
70     spline(function.size(), 0, 0, 0, 0, function.X(), function.Y(), factors.B(),
71     factors.C(), factors.D(), &flag);
72     if (flag != 0)
73     {
74         std::cout << "Error occured in spline function.\n";
75         return 1;
76     }
77     std::cout << "Spline coefficients:\n" << factors << "\n";
78
79     function_t newFunction(function.size());
80     std::copy(function.x.begin(), function.x.end(), newFunction.x.begin());
81     int last = 0;
82     std::transform(newFunction.x.begin(), newFunction.x.end(), newFunction.y.begin(),
83     [&](double x){
84         return seval(function.size(), x,
85         function.X(), function.Y(),
86         factors.B(), factors.C(), factors.D(), &last) -

```

```

86         RightPart(x);
87     });
88
89     std::cout << "Function given by equation f(x) = rightPart(x)\n" << newFunction << "\n";
90
91     std::vector<double> ans;
92     for (int i = 1; i < newFunction.size(); ++i)
93     {
94         if (newFunction.y[i] * newFunction.y[i - 1] < 0)
95         {
96             ans.push_back(bisection(function, factors,
97                                     newFunction.x[i - 1], newFunction.x[i], RightPart));
98         }
99     }
100
101     std::cout << "Number of roots: " << ans.size() << "\n";
102     std::copy(ans.begin(), ans.end(), std::ostream_iterator<double>(std::cout, " "));
103     std::cout << "\n";
104
105     return 0;
106 }

```

3.3 Код lab1_impl.cpp

```

1  #include "lab1.hpp"
2
3  #include <iostream>
4  #include <iterator>
5  #include <algorithm>
6  #include <functional>
7
8  std::istream& operator>>(std::istream& in, point_t& point)
9  {
10     in >> point.x >> point.y;
11     return in;
12 }
13 std::ostream& operator<<(std::ostream& out, const point_t& point)
14 {
15     out << point.x << " ; " << point.y << "\n";
16     return out;
17 }
18 function_t::function_t()
19 {}
20 function_t::function_t(size_t n):
21     x(n),
22     y(n)
23 {}
24 size_t function_t::size() const
25 {
26     return x.size();
27 }
28 double* function_t::X()
29 {
30     return x.data();
31 }
32 double* function_t::Y()
33 {
34     return y.data();
35 }
36
37 std::istream& operator>>(std::istream& in, function_t& function)

```

```

38 {
39     std::vector<point_t> points(std::istream_iterator<point_t>(std::ref(in)),
40                               std::istream_iterator<point_t>());
41     std::transform(points.begin(), points.end(), std::back_inserter(function.x), [](
42         const point_t& point){return point.x;});
43     std::transform(points.begin(), points.end(), std::back_inserter(function.y), [](
44         const point_t& point){return point.y;});
45     return in;
46 }
47 std::ostream& operator<<(std::ostream& out, const function_t& function)
48 {
49     std::transform(function.x.begin(), function.x.end(), function.y.begin(),
50                     std::ostream_iterator<point_t>(out), [](double x, double y){return point_t{x, y
51                     };});
52     return out;
53 }
54 spline_t::spline_t(size_t n):
55     b(n),
56     c(n),
57     d(n)
58 {
59 }
60 double* spline_t::B()
61 {
62     return b.data();
63 }
64 double* spline_t::C()
65 {
66     return c.data();
67 }
68 double* spline_t::D()
69 {
70     return d.data();
71 }
72 size_t spline_t::size() const
73 {
74     return b.size();
75 }
76 std::ostream& operator<<(std::ostream& out, const spline_t& factors)
77 {
78     for (int i = 0; i < factors.size(); ++i)
79     {
80         out << "spline " << i << ": " << factors.b[i] << "; " << factors.c[i] << "; " <<
            factors.d[i] << "\n";
81     }
82     return out;
83 }

```