

**Московский авиационный институт
(национальный исследовательский университет)**

**Институт №8 «Информационные технологии и прикладная
математика»**

Кафедра 806 «Вычислительная математика и программирование»

Лабораторные работы по курсу «Численные методы»

Студент: Г. С. Будайчиев
Преподаватель: Д. Е. Пивоваров
Группа: М8О-303Б-21
Дата:
Оценка:
Подпись:

Москва, 2024

3.1

1 Постановка задачи

Используя таблицу значений Y_i функции $y = f(x)$, вычисленных в точках $X_i, i = 0, \dots, 3$ построить интерполяционные многочлены Лагранжа и Ньютона, проходящие через точки $\{X_i, Y_i\}$. Вычислить значение погрешности интерполяции в точке X^* .

Вариант: 3

$$y = \tan(x), a) X_i = 0, \frac{\pi}{8}, \frac{2\pi}{8}, \frac{3\pi}{8}; X_i = 0, \frac{\pi}{8}, \frac{\pi}{3}, \frac{3\pi}{8}; X^* = \frac{3\pi}{16}$$

2 Результаты работы

Lagrange Method			
A	Ans: 0.644607	F(X*): 0.668179	Diff: 0.0235719
B	Ans: 0.585251	F(X*): 0.668179	Diff: 0.0829278
Newton Method			
A	Ans: 0.644607	F(X*): 0.668179	Diff: 0.0235719
B	Ans: 0.585251	F(X*): 0.668179	Diff: 0.0829278

Рис. 1: Вывод программы в консоли

3 Исходный код

```
1 | #include <iostream>
2 | #include <vector>
3 | #include <utility>
4 | #include <cmath>
5 |
6 | using namespace std;
7 |
8 | double F(double x)
9 | {
10 |     return tan(x);
11 | }
12 |
13 | double Lagrange(vector<double> X, vector<double> Y, double spec)
14 | {
15 |
16 |     double sum = 0;
17 |
18 |     for (int i = 0; i < X.size(); ++i)
19 |     {
20 |         double mult = 1;
21 |
22 |         for (int j = 0; j < X.size(); ++j)
23 |         {
24 |             if (i == j)
25 |                 continue;
26 |
27 |             mult = mult * ((spec - X[j]) / (X[i] - X[j]));
28 |         }
29 |
30 |         sum = sum + mult * Y[i];
31 |     }
32 |
33 |     return sum;
34 | }
35 |
36 | double Diff(vector<double> X, vector<double> Y, int n)
37 | {
38 |     switch (n)
39 |     {
40 |         case 0:
41 |             return Y[0];
42 |             break;
43 |
44 |         case 1:
45 |             return (Y[0] - Y[1]) / (X[0] - X[1]);
46 |             break;
47 |
```

```

48     default:
49         pair<vector<double>, vector<double>> X_v;
50         X_v.first.resize(n);
51         X_v.second.resize(n);
52
53         pair<vector<double>, vector<double>> Y_v;
54         Y_v.first.resize(n);
55         Y_v.second.resize(n);
56
57         for (int i = 0; i < n; ++i)
58         {
59             X_v.first[i] = X[i];
60             X_v.second[i] = X[i + 1];
61             Y_v.first[i] = Y[i];
62             Y_v.second[i] = Y[i + 1];
63         }
64
65         return (Diff(X_v.first, Y_v.first, n - 1) - Diff(X_v.second, Y_v.second, n
66                 - 1)) / (X[0] - X[n]);
67         break;
68     }
69 }
70 double Newton(vector<double> X, vector<double> Y, double spec)
71 {
72     vector<double> X_n(X.size());
73     vector<double> Y_n(Y.size());
74
75     double sum = 0;
76
77     for (int i = 0; i < X.size(); i++)
78     {
79         X_n[i] = X[i];
80         Y_n[i] = Y[i];
81     }
82
83     for (int i = 0; i < X_n.size(); ++i)
84     {
85         double mult = Diff(X_n, Y_n, i);
86
87         for (int j = 0; j < i; ++j)
88         {
89             mult = mult * (spec - X[j]);
90         }
91
92         sum = sum + mult;
93     }
94
95     return sum;

```

```

96 }
97
98
99 int main()
100 {
101     vector<double> X_a{ 0, M_PI / 8, M_PI / 4, 3 * M_PI / 8 };
102     vector<double> X_b{ 0, M_PI / 8, M_PI / 3, 3 * M_PI / 8 };
103
104     vector<double> Y_a;
105     vector<double> Y_b;
106
107     for (int i = 0; i < X_a.size(); i++)
108     {
109         Y_a.push_back(F(X_a[i]));
110         Y_b.push_back(F(X_b[i]));
111     }
112
113     double spec = 3 * M_PI / 16;
114
115     cout << "_____Lagrange Method_____"
116           << endl;
117
118     double A = Lagrange(X_a, Y_a, spec);
119     cout << "A | Ans: " << A << "\t\tF(X*): " << F(spec) << "\t\tDiff: " << abs(F(spec)
120           - A) << endl;
121
122     double B = Lagrange(X_b, Y_b, spec);
123     cout << "B | Ans: " << B << "\t\tF(X*): " << F(spec) << "\t\tDiff: " << abs(F(spec)
124           - B) << endl;
125
126     cout << "\n_____Newton
127           Method_____" << endl;
128
129     A = Newton(X_a, Y_a, spec);
130     cout << "A | Ans: " << A << "\t\tF(X*): " << F(spec) << "\t\tDiff: " << abs(F(spec)
131           - A) << endl;
132
133     B = Newton(X_b, Y_b, spec);
134     cout << "B | Ans: " << B << "\t\tF(X*): " << F(spec) << "\t\tDiff: " << abs(F(spec)
135           - B) << endl;
136 }

```

3.2

4 Постановка задачи

Построить кубический сплайн для функции, заданной в узлах интерполяции, предполагая, что сплайн имеет нулевую кривизну при $x = x_0$ и $x = x_4$. Вычислить значение функции в точке $x = X^*$.

Вариант: 3

$X_i = 1.5$					
i	0	1	2	3	4
x_i	0.0	0.9	1.8	2.7	3.6
f_i	0.0	0.36892	0.85408	1.7856	6.3138

Рис. 2: Условие

5 Результаты работы

```
____FUNC inc.____  
  
F(X*): 0.724543
```

Рис. 3: Вывод программы в консоли

6 Исходный код

```
1 | #include <iostream>
2 | #include <vector>
3 | #include <cmath>
4 |
5 | using namespace std;
6 |
7 | void FUNC(vector<double> X, vector<double> Y, double spec)
8 | {
9 |     vector<double> H(4);
10 |    for (int i = 0; i < 4; ++i)
11 |    {
12 |        H[i] = X[i + 1] - X[i];
13 |    }
14 |
15 |    vector<vector<double>> A(3);
16 |    for (int i = 0; i < A.size(); i++)
17 |    {
18 |        A[i].resize(3);
19 |    }
20 |
21 |    A[0][0] = 0;
22 |    A[0][1] = 2 * (H[0] + H[1]);
23 |    A[0][2] = H[1];
24 |
25 |    vector<double> B(4);
26 |    B[0] = 3 * ((Y[2] - Y[1]) / H[1] - (Y[1] - Y[0]) / H[0]);
27 |
28 |    for (int i = 1; i < 2; ++i)
29 |    {
30 |        A[i][0] = H[i];
31 |        A[i][1] = 2 * (H[i] + H[i + 1]);
32 |        A[i][2] = H[i + 1];
33 |        B[i] = 3 * ((Y[i + 2] - Y[i + 1]) / H[i + 1] - (Y[i + 1] - Y[i]) / H[i]);
34 |    }
35 |
36 |    A[2][0] = H[2];
37 |    A[2][1] = 2 * (H[2] + H[3]);
38 |    A[2][2] = 0;
39 |    B[2] = 3 * ((Y[4] - Y[3]) / H[3] - (Y[3] - Y[2]) / H[2]);
40 |
41 |    vector<double> P(3), C(4), Q(3);
42 |
43 |    Q[0] = B[0] / A[0][1];
44 |    P[0] = -A[0][2] / A[0][1];
45 |
46 |    for (int i = 1; i < 3; ++i)
47 |    {
```

```

48     P[i] = -A[i][2] / (A[i][1] + A[i][0] * P[i - 1]);
49     Q[i] = (B[i] - A[i][0] * Q[i - 1]) / (A[i][1] + A[i][0] * P[i - 1]);
50 }
51
52 C[0] = 0;
53 C[3] = Q[2];
54 for (int i = 2; i > 0; --i)
55 {
56     C[i] = P[i - 1] * C[i + 1] + Q[i - 1];
57 }
58
59 vector<double> a(4), b(4), d(4);
60 for (int i = 0; i < 3; ++i)
61 {
62     a[i] = Y[i];
63     b[i] = (Y[i + 1] - Y[i]) / H[i] - H[i] * (C[i + 1] + 2 * C[i]) / 3;
64     d[i] = (C[i + 1] - C[i]) / 3 / H[i];
65 }
66
67 a[3] = Y[3];
68 b[3] = (Y[4] - Y[3]) / H[3] - 2 / 3 * H[3] * C[3];
69 d[3] = -C[3] / 3 / H[3];
70
71 int i = 0;
72
73 while (X[i] < spec && X[i + 1] < spec)
74 {
75     i += 1;
76 }
77
78 cout << "___FUNC inc.___" << endl << endl << "F(X*): " << a[i] + b[i] * (spec - X[i]
79     ] + C[i] * pow(spec - X[i], 2) + d[i] * pow(spec - X[i], 3) << endl << "
80     -----";
81 }
82
83 int main()
84 {
85     vector<double> X{ 0.0, 0.9, 1.8, 2.7, 3.6 };
86     vector<double> Y{ 0.0, 0.36892, 0.85408, 1.7856, 6.3138 };
87
88     double spec = 1.5;
89     FUNC(X, Y, spec);
90 }

```


3.3

7 Постановка задачи

Для таблично заданной функции путем решения нормальной системы МНК найти приближающие многочлены а) 1-ой и б) 2-ой степени. Для каждого из приближающих многочленов вычислить сумму квадратов ошибок. Построить графики приближаемой функции и приближающих многочленов.

Вариант: 3

i	0	1	2	3	4	5
x_i	-0.9	0.0	0.9	1.8	2.7	3.6
y_i	-0.36892	0.0	0.36892	0.85408	1.7856	6.3138

Рис. 4: Условия

8 Результаты работы

FUNC inc.			
A	Coeff: -0.190134	1.24621	Error: 0.805786
B	Coeff: -0.464501	-0.125625	0.508086
			Error: 0.580048

Рис. 5: Вывод программы в консоли

9 Исходный код

```
1 | #include <iostream>
2 | #include <vector>
3 | #include <cmath>
4 |
5 | using namespace std;
6 |
7 | pair<vector<vector<double>>, vector<vector<double>>> createLU(vector<vector<double>> A
8 | )
9 | {
10 |     vector<vector<double>> U = A;
11 |     vector<vector<double>> L(A.size());
12 |
13 |     for (auto& elem : L)
14 |     {
15 |         for (int i = 0; i < A.size(); i++)
16 |         {
17 |             elem.push_back(0.0);
18 |         }
19 |     }
20 |
21 |     for (int k = 0; k < U.size(); ++k)
22 |     {
23 |         for (int i = k; i < U.size(); ++i)
24 |         {
25 |             L[i][k] = U[i][k] / U[k][k];
26 |         }
27 |
28 |         for (int i = k + 1; i < U.size(); ++i)
29 |         {
30 |             for (int j = k; j < U.size(); ++j)
31 |             {
32 |                 U[i][j] = U[i][j] - L[i][k] * U[k][j];
33 |             }
34 |         }
35 |     }
36 |
37 |     return make_pair(L, U);
38 | }
39 |
40 | vector<double> solve(vector<vector<double>> A, vector<double> B)
41 | {
42 |     pair<vector<vector<double>>, vector<vector<double>>> LU = createLU(A);
43 |
44 |     vector<vector<double>> L = LU.first;
45 |     vector<vector<double>> U = LU.second;
46 |
```

```

47     vector<double> Z(A.size());
48
49     Z[0] = B[0];
50
51     for (int i = 1; i < A.size(); ++i)
52     {
53         double sum = 0;
54
55         for (int j = 0; j < i; ++j)
56         {
57             sum += L[i][j] * Z[j];
58         }
59
60         Z[i] = B[i] - sum;
61     }
62
63     vector<double> x(A.size());
64
65     x[A.size() - 1] = Z[A.size() - 1] / U[A.size() - 1][A.size() - 1];
66
67     for (int i = A.size() - 2; i >= 0; --i)
68     {
69         double sum = 0;
70
71         for (int j = i + 1; j < A.size(); ++j)
72         {
73             sum += U[i][j] * x[j];
74         }
75
76         x[i] = (Z[i] - sum) / U[i][i];
77     }
78
79     return x;
80 }
81
82 vector<double> FUNC(vector<double> X, vector<double> Y, int m)
83 {
84     vector<vector<double>> A(m + 1, vector<double>(m + 1));
85
86     vector<double> b(m + 1);
87
88     for (int k = 0; k < m + 1; ++k)
89     {
90         for (int j = 0; j < m + 1; ++j)
91         {
92             double sum = 0;
93
94             for (int i = 0; i < X.size(); ++i)
95             {

```

```

96         sum += pow(X[i], k + j);
97     }
98
99     A[k][j] = sum;
100 }
101
102 double sum = 0;
103
104 for (int i = 0; i < X.size(); ++i)
105 {
106     sum += Y[i] * pow(X[i], k);
107 }
108
109 b[k] = sum;
110 }
111
112 vector<double> a = solve(A, b);
113
114 return a;
115 }
116
117 double error(vector<double> X, vector<double> Y, vector<double> A)
118 {
119     double s = 0;
120
121     for (int i = 0; i < A.size(); ++i)
122     {
123         s = s + (A[i] - Y[i+1]) * (A[i] - Y[i+1]);
124     }
125
126     return s;
127 }
128
129
130 int main()
131 {
132     vector<double> X{ -0.9, 0.0, 0.9, 1.8, 2.7, 3.6 };
133     vector<double> Y{ -0.36892, 0.0, 0.36892, 0.85408, 1.7856, 6.3138 };
134
135     vector<double> A = FUNC(X, Y, 1);
136
137     cout << "-----FUNC inc.-----" << endl
138           << "A | Coeff:";
139
140     for(auto& elem : A)
141     {
142         cout << " " << elem << " ";
143     }

```

```

144     cout << "\t\t\t| Error: " << error(X, Y, A) << endl;
145
146     A = FUNC(X, Y, 2);
147
148     cout << "B | Coeff:";
149
150     for (auto& elem : A)
151     {
152         cout << " " << elem << " ";
153     }
154
155     cout << "\t| Error: " << error(X, Y, A) << endl;
156
157     return 0;
158 }

```

3.4

10 Постановка задачи

Вычислить первую и вторую производную от таблично заданной функции $y_i = f(x_i)$, $i = 0, 1, 2, 3, 4$ в точке $x = X_i$.

Вариант: 3

$X^* = 2.0$

i	0	1	2	3	4
x_i	1.0	1.5	2.0	2.5	3.0
y_i	0.0	0.40547	0.69315	0.91629	1.0986

Рис. 6: Условия

11 Результаты работы

```
First derivative in X*(2),    f'(X*) = 0.69315
Second derivative in X*(2),   f''(X*) = -0.47116
```

Рис. 7: Вывод программы в консоли

12 Исходный код

```
1 #include <iostream>
2 #include <vector>
3 #include <cmath>
4
5 using namespace std;
6
7 bool leq(double a, double b)
8 {
9     return (a < b) || (abs(b - a) < 1e-9);
10 }
11
12 double Der1(vector<double> X, vector<double> Y, double spec)
13 {
14     for (int i = 0; i < X.size() - 1; ++i)
15     {
16         if (X[i] < spec && leq(spec, X[i + 1]))
17         {
18             double ANS = (Y[i + 1] - Y[i - 1]) / (X[i + 1] - X[i - 1]);
19             return ANS;
20         }
21     }
22     return 0;
23 }
24
25 double Der2(vector<double> X, vector<double> Y, double spec)
26 {
27     for (int i = 0; i < X.size() - 1; ++i)
28     {
29         if (X[i] < spec && leq(spec, X[i + 1]))
30         {
31             double ANS = (Y[i - 1] - 2 * Y[i] + Y[i + 1]) / pow((X[i + 1] - X[i]), 2);
32             return ANS;
33         }
34     }
35     return 0;
36 }
37
38
39 int main()
40 {
41     vector<double> X{ 1.0, 1.5, 2.0, 2.5, 3.0 };
42     vector<double> Y{ 0.0, 0.40547, 0.69315, 0.91629, 1.0986 };
43     double spec = 2.0;
44
45     cout << "First derivative in X*(" << spec << ")\tf'(X*) = " << Der1(X, Y, spec) <<
        endl;
```

```

46 || cout << "Second derivative in X*(" << spec << ")\tf\"(X*) = " << Der2(X, Y, spec)
47 || << endl;

```


3.5

13 Постановка задачи

Вычислить определенный интеграл $\int_{X_0}^{X_1} y dx$, методами прямоугольников, трапеций, Симпсона с шагами h_1, h_2 . Оценить погрешность вычислений, используя Метод Рунге-Ромберга:

Вариант: 3

$$y = \frac{x}{(3x+4)^3} \quad X_0 = -1, X_k = 1, h_1 = 0.5, h_2 = 0.25$$

14 Результаты работы

```

      _____H1_____
Rectangle: -0.0724458
Trapeze:  -0.265249
Simpson:  -0.137741
|
      _____H2_____
Rectangle: -0.103192
Trapeze:  -0.168088
Simpson:  -0.12559
|
      _____Error_____
Rect:  -0.133937
Trapeze: -0.135701
Simpson: -0.12478
|
```

Рис. 8: Вывод программы в консоли

15 Исходный код

```
1 | #include <iostream>
2 | #include <cmath>
3 |
4 | using namespace std;
5 |
6 | double Func(double x)
7 | {
8 |     return x / pow((3 * x + 4), 3);
9 | }
10 |
11 | double Rectangle(double X0, double Xk, double h)
12 | {
13 |     double sum = 0;
14 |     while (X0 + h < Xk)
15 |     {
16 |         sum += Func(X0 + h / 2);
17 |         X0 += h;
18 |     }
19 |
20 |     return sum * h;
21 | }
22 |
23 | double Trapeze(double X0, double Xk, double h)
24 | {
25 |     double sum = 0;
26 |
27 |     while (X0 + h < Xk)
28 |     {
29 |         sum += (Func(X0 + h) + Func(X0));
30 |         X0 += h;
31 |     }
32 |
33 |     return sum * h * 0.5;
34 | }
35 |
36 | double Simpson(double X0, double Xk, double h)
37 | {
38 |     double sum = 0;
39 |     X0 += h;
40 |     while (X0 + h < Xk)
41 |     {
42 |         sum += Func(X0 - h) + 4 * Func(X0 - h / 2) + Func(X0);
43 |         X0 += h;
44 |     }
45 |
46 |     return sum * h / 6;
47 | }
```

```

48
49 double rungeRombert(double h1, double h2, double i1, double i2, double p)
50 {
51     return i1 + (i1 - i2) / (pow((h2 / h1), p) - 1);
52 }
53
54 int main() {
55     double X0 = -1;
56     double Xk = 1;
57     double h1 = 0.5;
58     double h2 = 0.25;
59
60     cout << "_____H1_____" << endl;
61     cout << "Rectangle: " << Rectangle(X0, Xk, h1) << endl;
62     cout << "Trapeze: " << Trapeze(X0, Xk, h1) << endl;
63     cout << "Simpson: " << Simpson(X0, Xk, h1) << endl << endl;
64
65     cout << "_____H2_____" << endl;
66     cout << "Rectangle: " << Rectangle(X0, Xk, h2) << endl;
67     cout << "Trapeze: " << Trapeze(X0, Xk, h2) << endl;
68     cout << "Simpson: " << Simpson(X0, Xk, h2) << endl << endl;
69
70     cout << "_____Error_____" << endl;
71     cout << "Rect: " << rungeRombert(h1, h2, Rectangle(X0, Xk, h1), Rectangle(X0, Xk,
        h2), 1) << endl;
72     cout << "Trapeze: " << rungeRombert(h1, h2, Trapeze(X0, Xk, h1), Trapeze(X0, Xk, h2
        ), 2) << endl;
73     cout << "Simpson: " << rungeRombert(h1, h2, Simpson(X0, Xk, h1), Simpson(X0, Xk, h2
        ), 4) << endl;
74
75     return 0;
76 }

```