

**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: Первухин А.С.
Преподаватель: Пивоваров Д.Е.
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: 17

$$17. y = e^x + x, \quad \text{a) } X_i = -2, -1, 0, 1; \quad \text{б) } X_i = -2, -1, 0.2, 1; \quad X^* = -0.5.$$

2 Результаты работы

```
1  Коэффициенты Лагранжа:
2  0.310777
3  -0.31606
4  -0.5
5  0.619714
6  Значение многочлена Лагранжа: 0.0910811
7  Погрешность интерполяции в точке x: 0.0154495
8  -----
9  Коэффициенты Ньютона:
10 -1.86466
11  1.23254
12  0.217602
13  0.120771
14  Значение многочлена Ньютона: 0.0839487
15  Погрешность интерполяции в точке x: 0.022582
```

Рис. 1: Вывод программы

3 Исходный код

```
1 |
2 | #include <fstream>
3 | #include <vector>
4 | #include <iostream>
5 |
6 | using namespace std;
7 |
8 | int main()
9 | {
10 |     int n = 5;
```

```

11 | ofstream fout;
12 | fout.open("output.txt");
13 | vector<vector<double>> A = {
14 |     {-6, 5, 0, 0, 0},
15 |     {-1, 13, 6, 0, 0},
16 |     {0, -9, -15, -4, 0},
17 |     {0, 0, -1, -7, 1},
18 |     {0, 0, 0, 9, -18}
19 | };
20 | vector<double> d = { 51, 100, -12, 47, -90 };
21 | vector<double> P(n, 0);
22 | vector<double> Q(n, 0);
23 | vector<double> x(n, 0);
24 | P[0] = -A[0][1] / A[0][0];
25 | Q[0] = d[0] / A[0][0];
26 | for (int i = 1; i < n; i++) {
27 |     P[i] = -A[i][i + 1] / (A[i][i] + A[i][i - 1] * P[i - 1]);
28 |     Q[i] = (d[i] - A[i][i - 1] * Q[i - 1]) / (A[i][i] + A[i][i - 1] * P[i - 1]);
29 | }
30 | for (int i = n - 1; i >= 0; i--) {
31 |     x[i] = P[i] * x[i + 1] + Q[i];
32 | }
33 | for (int i = 0; i < n; i++) {
34 |     fout << x[i] << endl;
35 | }
36 | return 0;
37 | }

```

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: 17

17. $X^* = -0.5$

i	0	1	2	3	4
x_i	-2.0	-1.0	0.0	1.0	2.0
f_i	-1.8647	-0.63212	1.0	3.7183	9.3891

Рис. 2: Условие

5 Результаты работы

1 Значение функции в точке X: 0.244152

Рис. 3: Вывод программы

6 Исходный код

```
1 | #include <iostream>
2 | #include <vector>
3 | #include <fstream>
4 | #include <cmath>
5 | #include <iomanip>
6 |
7 | using namespace std;
8 |
9 | vector<double> SolveSys(vector<vector<double>>& A, vector<double>& d, int n){
10 |     vector<double> P (n, 0);
11 |     vector<double> Q (n, 0);
```

```

12     vector<double> x (n, 0);
13     P[0] = - A[0][1] / A[0][0];
14     Q[0] = d[0] / A[0][0];
15     for (int i = 1; i < n; i++){
16         P[i] = - A[i][i+1] / (A[i][i] + A[i][i-1] * P[i-1]);
17         Q[i] = (d[i] - A[i][i-1] * Q[i-1]) / (A[i][i] + A[i][i-1] * P[i-1]);
18     }
19     for (int i = n - 1; i >= 0; i--){
20         x[i] = P[i] * x[i+1] + Q[i];
21     }
22     return x;
23 }
24
25 double Spline(vector<double>& x, vector<double>& f, double x0, int n = 4){
26     double res;
27     vector<double> h (n+1,0);
28     vector<double> a (n+1,0);
29     vector<double> b (n+1,0);
30     vector<double> c (n+1,0);
31     vector<double> d (n+1,0);
32     for (int i = 1; i <= n; i++){
33         h[i] = x[i] - x[i-1];
34     }
35     vector<vector<double>> A = {
36         {2*(h[1] + h[2]), h[2], 0},
37         {h[2], 2*(h[2] + h[3]), h[3]},
38         {0, h[3], 2*(h[3] + h[4])}
39     };
40     vector<double> B = {3*((f[2] - f[1])/h[2] - (f[1] - f[0])/h[1]),
41         3*((f[3] - f[2])/h[3] - (f[2] - f[1])/h[2]),
42         3*((f[4] - f[3])/h[4] - (f[3] - f[2])/h[3])};
43     vector<double> c_0 = SolveSys(A, B, n-1);
44     c[1] = 0;
45     c[2] = c_0[0];
46     c[3] = c_0[1];
47     c[4] = c_0[2];
48     for (int i = 1; i <= n; i++){
49         a[i] = f[i-1];
50     }
51     for (int i = 1; i <= n-1; i++){
52         b[i] = (f[i] - f[i-1])/h[i] - 1/3 * h[i]*(c[i+1] + 2*c[i]);
53         d[i] = (c[i+1] - c[i])/(3*h[i]);
54     }
55     b[n] = (f[n] - f[n-1])/h[n] - 2/3*h[n]*c[n];
56     d[n] = -c[n]/(3*h[n]);
57     res = a[2] + b[2]*(x0 - x[1]) + c[2]*pow(x0 - x[1],2) + d[2]*pow(x0 - x[1],3);
58
59     return res;
60 }

```

```

61 |
62 | int main() {
63 |     ofstream fout;
64 |     fout.open("output.txt");
65 |     vector<double> x = {-2, -1, 0, 1, 2};
66 |     vector<double> f = {-1.8647, -0.63212, 1, 3.7183, 9.3891};
67 |     double X0 = -0.5;
68 |     fout << "      X: " << Spline(x, f, X0) << endl;
69 |     return 0;
70 | }

```

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: 17

17.

i	0	1	2	3	4	5
x_i	-3.0	-2.0	-1.0	0.0	1.0	2.0
y_i	-2.9502	-1.8647	-0.63212	1.0	3.7183	9.3891

Рис. 4: Условия

8 Результаты работы

Приближающие многочлены 1 степени:

2.58736 2.28793

Сумма квадратов ошибок: 11.4549

Приближающие многочлены 2 степени:

3.30687 3.72694 0.93269

Сумма квадратов ошибок: 64.8558

Рис. 5: Вывод программы

9 Исходный код

```
1 | #include <iostream>
2 | #include <vector>
3 | #include <fstream>
4 | #include <cmath>
5 |
```

```

6
7 using namespace std;
8
9 vector<double> SolveSys(vector<vector<double>>& A, vector<double>& d, int n){
10     vector<double> P (n, 0);
11     vector<double> Q (n, 0);
12     vector<double> x (n, 0);
13     P[0] = - A[0][1] / A[0][0];
14     Q[0] = d[0] / A[0][0];
15     for (int i = 1; i < n; i++){
16         P[i] = - A[i][i+1] / (A[i][i] + A[i][i-1] * P[i-1]);
17         Q[i] = (d[i] - A[i][i-1] * Q[i-1]) / (A[i][i] + A[i][i-1] * P[i-1]);
18     }
19     for (int i = n - 1; i >= 0; i--){
20         x[i] = P[i] * x[i+1] + Q[i];
21     }
22     return x;
23 }
24
25
26
27 tuple<vector<double>, vector<double>, double, double> MNK(vector<double>& x, vector<
28     double>& y, int n){
29     double xsum = 0, ysum = 0, xsum2 = 0, ysum2 = 0, xysum = 0, xsum3 = 0, xsum4 = 0,
30         xysum2 = 0;
31
32     for (int i = 0; i <= n; i++){
33         xsum += x[i];
34         ysum += y[i];
35         xsum2 += x[i] * x[i];
36         ysum2 += y[i] * y[i];
37         xysum += x[i] * y[i];
38         xsum3 += pow(x[i], 3);
39         xsum4 += pow(x[i], 4);
40         xysum2 += x[i] * y[i] * x[i];
41     }
42     vector<vector<double>> A1= {
43         {(double)(n+1), xsum},
44         {xsum, xsum2}
45     };
46     vector<double> B1 = {ysum, xysum};
47     vector<double> coef1 = SolveSys(A1, B1, 2);
48     vector<vector<double>> A2= {
49         {(double)(n+1), xsum, xsum2},
50         {xsum, xsum2, xsum3},
51         {xsum2, xsum3, xsum4}
52     };
53     vector<double> B2 = {ysum, xysum, xysum2};

```



```

53     vector<double> coef2 = SolveSys(A2, B2, 3);
54
55     vector<double> f (n+1, 0);
56     double err1 = 0;
57     for (int i = 0; i <= n; i++){
58         f[i] = coef1[0] + coef1[1] * x[i];
59     }
60     for (int i = 0; i <= n; i++){
61         err1 += pow(f[i] - y[i], 2);
62     }
63
64     double err2 = 0;
65     for (int i = 0; i <= n; i++){
66         f[i] = coef2[0] + coef2[1] * x[i] + coef2[2] * x[i] * x[i];
67     }
68     for (int i = 0; i <= n; i++){
69         err2 += pow(f[i] - y[i], 2);
70     }
71
72     return make_tuple(coef1, coef2, err1, err2);
73 }
74
75 int main() {
76     ofstream fout;
77     fout.open("output.txt");
78     int n = 5;
79     vector<double> x = {-3, -2, -1, 0, 1, 2};
80     vector<double> y = {-2.9502, -1.8647, -0.63212, 1, 3.7183, 9.3891};
81     auto[coefs1, coefs2, err1, err2] = MNK(x, y, n);
82     fout << " 1 :" << endl;
83     for (int i = 0; i < 2; i++){
84         fout << coefs1[i] << " ";
85     }
86     fout << endl;
87     fout << "C : " << err1 << endl;
88     fout << endl;
89     fout << " 2 :" << endl;
90     for (int i = 0; i < 3; i++){
91         fout << coefs2[i] << " ";
92     }
93     fout << endl;
94     fout << "C : " << err2 << endl;
95     return 0;
96 }

```

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: 17

17. $X^* = 0.2$

i	0	1	2	3	4
x_i	-0.2	0.0	0.2	0.4	0.6
y_i	-0.40136	0.0	0.40136	0.81152	1.2435

Рис. 6: Условия

11 Результаты работы

Первая производная в точке X: 2.0288

Вторая производная в точке X: 0.22

Рис. 7: Вывод программы

12 Исходный код

```
1 | #include <iostream>
2 | #include <vector>
3 | #include <fstream>
4 |
5 | using namespace std;
6 |
7 | double Diff1(vector<double>& x, vector<double>& y, double x0, int i){
8 |     double res;
9 |     res = (y[i+1] - y[i])/(x[i+1] - x[i]) + (((y[i+2] - y[i+1])/(x[i+2] - x[i+1]) - (y[
10 |         i+1] - y[i])/(x[i+1] - x[i]))/(x[i+2] - x[i])) * (2*x0 - x[i] - x[i+1]));
11 |     return res;
12 | }
13 | double Diff2(vector<double>& x, vector<double>& y, double x0, int i){
14 |     double res;
```

```

15     res = 2 * (((y[i+2] - y[i+1])/(x[i+2] - x[i+1]) - (y[i+1] - y[i])/(x[i+1] - x[i]))
16           /(x[i+2] - x[i]));
17     return res;
18 }
19 int main() {
20     ofstream fout("output.txt");
21     vector<double> x = {-0.2, 0, 0.2, 0.4, 0.6};
22     vector<double> y = {-0.40136, 0.0, 0.40136, 0.81152, 1.2435};
23     double x0 = 0.2;
24     int p = 0;
25     for (int i = 0; i < 5; i++){
26         if (x0 >= x[i] && x0 <= x[i+1]){
27             p = i;
28             break;
29         }
30     }
31     double res1 = Diff1(x, y, x0, p);
32     double res2 = Diff2(x, y, x0, p);
33     fout << "      : " << res1 << endl;
34     fout << "      : " << res2 << endl;
35     return 0;
36 }

```

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: 17

$$17. \quad y = \frac{1}{256 - x^4}, \quad X_0 = -2, \quad X_k = 2, \quad h_1 = 1.0, \quad h_2 = 0.5;$$

14 Результаты работы

```
----- Метод прямоугольников -----  
Результат с шагом 1: 0.00794138  
Результат с шагом 0.5: 0.00797292  
Погрешность: 0.00793087  
----- Метод трапеций -----  
Результат с шагом 1: 0.00784314  
Результат с шагом 0.5: 0.00777737  
Погрешность: 0.00786506  
----- Метод Симпсона -----  
Результат с шагом 1: 0.00790863  
Результат с шагом 0.5: 0.00790774  
Погрешность: 0.00790893
```

Рис. 8: Вывод программы

15 Исходный код

```
1  #include <iostream>
2  #include <cmath>
3  #include <vector>
4  #include <fstream>
5
6  using namespace std;
7
8  double Y(double x){
9      return 1/(256 - pow(x,4));
10 }
11
12 pair <double, double> Rectangle(double x0, double xk, double h1, double h2){
13     int n1 = 0, n2 = 0;
14     double x = x0;
15     double x1 = x0;
16     double res = 0, res1 = 0;
17     double h = 0;
18     vector <double> xi1, xi2;
19     while (x != xk){
20         xi1.push_back(x);
21         x += h1;
22         n1++;
23     }
24     n1++;
25     xi1.push_back(x);
26     while (x1 != xk){
27         xi2.push_back(x1);
28         x1 += h2;
29         n2++;
30     }
31     n2++;
32     xi2.push_back(x1);
33
34     for (int i = 1; i <= n1; i++){
35         h = xi1[i] - xi1[i-1];
36         res += h * Y((xi1[i] + xi1[i-1]) / 2);
37     }
38
39     for (int i = 1; i <= n2; i++){
40         h = xi2[i] - xi2[i-1];
41         res1 += h * Y((xi2[i] + xi2[i-1]) / 2);
42     }
43     return make_pair(res, res1);
44 }
45
46 pair <double, double> Trapez(double x0, double xk, double h1, double h2){
47     int n1 = 0, n2 = 0;
```

```

48     double x = x0;
49     double x1 = x0;
50     double res = 0, res1 = 0;
51     double h = 0;
52     vector <double> xi1, xi2;
53     while (x != xk){
54         xi1.push_back(x);
55         x += h1;
56         n1++;
57     }
58     n1++;
59     xi1.push_back(x);
60     while (x1 != xk){
61         xi2.push_back(x1);
62         x1 += h2;
63         n2++;
64     }
65     n2++;
66     xi2.push_back(x1);
67
68     for (int i = 1; i <= n1; i++){
69         h = xi1[i] - xi1[i-1];
70         res += h * (Y(xi1[i]) + Y(xi1[i-1]));
71     }
72     res *= 0.5;
73     for (int i = 1; i <= n2; i++){
74         h = xi2[i] - xi2[i-1];
75         res1 += h * (Y(xi2[i]) + Y(xi2[i-1]));
76     }
77     res1 *= 0.5;
78     return make_pair(res, res1);
79 }
80
81 pair <double, double> Simpson(double x0, double xk, double h1, double h2){
82     int n1 = 0, n2 = 0;
83     double x = x0;
84     double x1 = x0;
85     double res = 0, res1 = 0;
86     double h = 0;
87     vector <double> xi1, xi2;
88     while (x != xk){
89         xi1.push_back(x);
90         x += h1;
91         n1++;
92     }
93     n1++;
94     xi1.push_back(x);
95     while (x1 != xk){
96         xi2.push_back(x1);

```

```

97         x1 += h2;
98         n2++;
99     }
100     n2++;
101     xi2.push_back(x1);
102
103     for (int i = 1; i <= n1; i++){
104         h = (xi1[i] - xi1[i-1]) / 2;
105         res += h * (Y(xi1[i-1]) + 4 * Y((xi1[i-1] + xi1[i])/2) + Y(xi1[i]));
106     }
107     res = res / 3;
108     for (int i = 1; i <= n2; i++){
109         h = (xi2[i] - xi2[i-1]) / 2;
110         res1 += h * (Y(xi2[i-1]) + 4 * Y((xi2[i-1] + xi2[i])/2) + Y(xi2[i]));
111     }
112     res1 = res1 / 3;
113     return make_pair(res, res1);
114 }
115
116 double Runge(double F1, double F2, double p){
117     return F1 + (F1 - F2)/(pow(2, p) - 1);
118 }
119
120 int main() {
121     ofstream fout("output.txt");
122     double x0 = -2;
123     double xk = 2;
124     double h1 = 1;
125     double h2 = 0.5;
126     double res1 = Rectangle(x0, xk, h1, h2).first, res2 = Rectangle(x0, xk, h1, h2).
        second;
127     fout << "-----" << endl;
128     fout << "    " << h1 << ": " << res1 << endl << "    " << h2 << ": " << res2 << endl;
129     fout << ": " << Runge(res1, res2, 2) << endl;
130     fout << "-----" << endl;
131     res1 = Trapez(x0, xk, h1, h2).first;
132     res2 = Trapez(x0, xk, h1, h2).second;
133     fout << "    " << h1 << ": " << res1 << endl << "    " << h2 << ": " << res2 << endl;
134     fout << ": " << Runge(res1, res2, 2) << endl;
135     fout << "-----" << endl;
136     res1 = Simpson(x0, xk, h1, h2).first;
137     res2 = Simpson(x0, xk, h1, h2).second;
138     fout << "    " << h1 << ": " << res1 << endl << "    " << h2 << ": " << res2 << endl;
139     fout << ": " << Runge(res1, res2, 2) << endl;
140     return 0;
141 }

```