

**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: Батулин Е.А.
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: 1

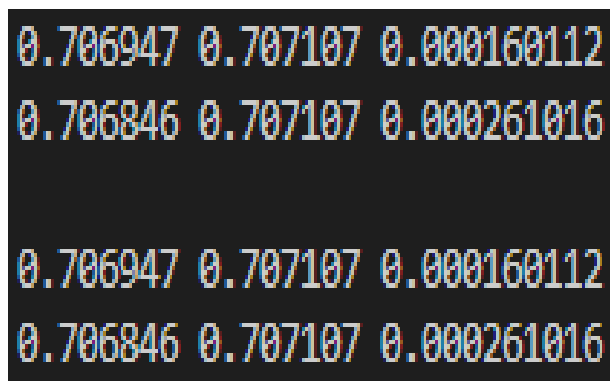
$y = \sin x$,

a) $X_i = 0.1\pi, 0.2\pi, 0.3\pi, 0.4\pi$;

b) $X_i = 0.1\pi, \pi/6, 0.3\pi, 0.4\pi$;

$X^* = \pi/4$

2 Результаты работы



```
0.706947 0.707107 0.000160112
0.706846 0.707107 0.000261016

0.706947 0.707107 0.000160112
0.706846 0.707107 0.000261016
```

Рис. 1: Вывод программы в консоли

3 Исходный код

```
1 | #define _USE_MATH_DEFINES
2 |
3 | #include <iostream>
4 | #include <vector>
5 | #include <string>
6 | #include <cmath>
7 |
8 | using namespace std;
9 |
10 | double initFunc(double x){
11 |     return sin(x);
12 | }
13 |
14 | double Lagrange(double X[], double Y[], int n, double x){
15 |     double s = 0;
16 |     for (int i = 0; i < n; ++i){
17 |         double p = 1;
18 |         for (int j = 0; j < n; ++j){
19 |             if (i != j){
20 |                 p *= (x - X[j]) / (X[i] - X[j]);
21 |             }
22 |         }
23 |         s += Y[i] * p;
24 |     }
25 |     return s;
26 | }
27 |
28 | double SplitDifference(double X[], double Y[], int n){
29 |     if (n == 0){
30 |         return Y[0];
31 |     }
32 |     else if (n == 1){
33 |         return (Y[0] - Y[1]) / (X[0] - X[1]);
34 |     }
35 |     else{
36 |         double X_a[n];
37 |         double X_b[n];
38 |         double Y_a[n];
39 |         double Y_b[n];
40 |         for (int i = 0; i < n; ++i){
41 |             X_a[i] = X[i];
42 |             X_b[i] = X[i + 1];
43 |             Y_a[i] = Y[i];
44 |             Y_b[i] = Y[i + 1];
45 |         }
46 |         return (SplitDifference(X_a, Y_a, n - 1) - SplitDifference(X_b, Y_b, n - 1)) /
                (X[0] - X[n]);
```

```

47     }
48 }
49
50 double Newton(double X[], double Y[], int n, double x){
51     double s = 0;
52     double X_n[n];
53     double Y_n[n];
54     for (int i = 0; i < n; ++i){
55         X_n[i] = X[i];
56         Y_n[i] = Y[i];
57         double p = SplitDifference(X_n, Y_n, i);
58         for (int j = 0; j < i; ++j){
59             p *= (x - X[j]);
60         }
61         s += p;
62     }
63     return s;
64 }
65
66
67 int main(){
68     int n = 4;
69     double X_1[n] = {0.1 * M_PI, 0.2 * M_PI, 0.3 * M_PI, 0.4 * M_PI};
70     double X_2[n] = {0.1 * M_PI, M_PI / 6, 0.3 * M_PI, 0.4 * M_PI};
71     double Y_1[n] = {initFunc(X_1[0]), initFunc(X_1[1]), initFunc(X_1[2]), initFunc(X_1
72         [3])};
73     double Y_2[n] = {initFunc(X_2[0]), initFunc(X_2[1]), initFunc(X_2[2]), initFunc(X_2
74         [3])};
75     double x = M_PI / 4;
76
77     double L_1 = Lagrange(X_1, Y_1, n, x);
78     cout << L_1 << " " << initFunc(x) << " " << abs(L_1 - initFunc(x)) << "\n";
79     double L_2 = Lagrange(X_2, Y_2, n, x);
80     cout << L_2 << " " << initFunc(x) << " " << abs(L_2 - initFunc(x)) << "\n";
81     cout << "\n";
82
83     double N_1 = Newton(X_1, Y_1, n, x);
84     cout << N_1 << " " << initFunc(x) << " " << abs(N_1 - initFunc(x)) << "\n";
85     double N_2 = Newton(X_2, Y_2, n, x);
86     cout << N_2 << " " << initFunc(x) << " " << abs(N_2 - initFunc(x)) << "\n";
87 }

```

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: 1

1. $X^* = 1.5$

i	0	1	2	3	4
x_i	0.0	1.0	2.0	3.0	4.0
f_i	0.0	0.5	0.86603	1.0	0.86603

Рис. 2: Условия

5 Результаты работы



```
0.710467
```

Рис. 3: Вывод программы в консоли

6 Исходный код

```
1 | #include <iostream>
2 | #include <vector>
3 | #include <string>
4 | #include <cmath>
5 |
6 | using namespace std;
7 |
8 | class Matrix {
9 | public:
10 |     int rows, cols;
11 |     double **a;
12 |
13 |     Matrix (){
14 |         rows = 0;
15 |         cols = 0;
16 |         a = new double*[rows];
17 |         for(int i = 0; i < rows; i++)
18 |             a[i] = new double[cols];
19 |     }
20 |
21 |     Matrix(int n, int m, bool identity = 0)
22 |     {
23 |         rows = n;
24 |         cols = m;
25 |         a = new double *[rows];
26 |         for (int i = 0; i < rows; i++){
27 |             a[i] = new double[cols];
28 |             for (int j = 0; j < cols; j++){
29 |                 a[i][j] = identity * (i == j);
30 |             }
31 |         }
32 |     }
33 |
34 |     double* operator[](int i)
35 |     {
36 |         return a[i];
37 |     }
38 | };
39 |
40 | double Spline(double X[], double Y[], int n, double x){
41 |     double h[n];
42 |     for (int i = 0; i < n; ++i){
43 |         h[i] = X[i + 1] - X[i];
44 |     }
45 |
46 |     //
47 |     Matrix A = Matrix (n - 1, 3);
```

```

48     double B[n];
49
50     A[0][0] = 0;
51     A[0][1] = 2 * (h[0] + h[1]);
52     A[0][2] = h[1];
53     B[0] = 3 * ((Y[2] - Y[1]) / h[1] - (Y[1] - Y[0]) / h[0]);
54     for (int i = 1; i < n - 2; ++i){
55         A[i][0] = h[i];
56         A[i][1] = 2 * (h[i] + h[i + 1]);
57         A[i][2] = h[i + 1];
58         B[i] = 3 * ((Y[i + 2] - Y[i + 1]) / h[i + 1] - (Y[i + 1] - Y[i]) / h[i]);
59     }
60     A[n - 2][0] = h[n - 2];
61     A[n - 2][1] = 2 * (h[n - 2] + h[n - 1]);
62     A[n - 2][2] = 0;
63     B[n - 2] = 3 * ((Y[n] - Y[n - 1]) / h[n - 1] - (Y[n - 1] - Y[n - 2]) / h[n - 2]);
64
65     //
66     double P[n - 1];
67     double Q[n - 1];
68     double c[n];
69     P[0] = -A[0][2] / A[0][1];
70     Q[0] = B[0] / A[0][1];
71     for (int i = 1; i < n - 1; ++i){
72         P[i] = -A[i][2] / (A[i][1] + A[i][0] * P[i - 1]);
73         Q[i] = (B[i] - A[i][0] * Q[i - 1]) / (A[i][1] + A[i][0] * P[i - 1]);
74     }
75
76     //
77     c[0] = 0;
78     c[n - 1] = Q[n - 1];
79     for (int i = n - 2; i > 0; --i){
80         c[i] = P[i - 1] * c[i + 1] + Q[i - 1];
81     }
82
83     double a[n], b[n], d[n];
84     for (int i = 0; i < n - 1; ++i){
85         a[i] = Y[i];
86         b[i] = (Y[i + 1] - Y[i]) / h[i] - h[i] * (c[i + 1] + 2 * c[i]) / 3;
87         d[i] = (c[i + 1] - c[i]) / 3 / h[i];
88     }
89     a[n - 1] = Y[n - 1];
90     b[n - 1] = (Y[n] - Y[n - 1]) / h[n - 1] - 2 / 3 * h[n - 1] * c[n - 1];
91     d[n - 1] = - c[n - 1] / 3 / h[n - 1];
92
93     int i = 0;
94     while (X[i] < x and X[i + 1] < x){
95         i += 1;
96     }

```

```

97     return a[i] + b[i] * (x - X[i]) + c[i] * pow(x - X[i], 2) + d[i] * pow(x - X[i], 3)
98     ;
99 }
100 int main()
101 {
102     int n = 5;
103     double X[n] = {0.0, 1.0, 2.0, 3.0, 4.0};
104     double Y[n] = {0.0, 0.5, 0.86603, 1.0, 0.86603};
105     double x = 1.5;
106
107     cout << Spline(X, Y, n - 1, x);
108 }

```


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: 1

1.

i	0	1	2	3	4	5
x_i	-1.0	0.0	1.0	2.0	3.0	4.0
y_i	-0.5	0.0	0.5	0.86603	1.0	0.86603

Рис. 4: Условия

8 Результаты работы

```
0.270818 0.0151789 0.000346381
```

Рис. 5: Вывод программы в консоли

9 Исходный код

```
1 | #include <iostream>
2 | #include <vector>
3 | #include <string>
4 | #include <cmath>
5 |
6 | using namespace std;
7 |
8 | class Matrix {
9 |     public:
10 |     int rows, cols;
11 |     double **a;
12 |
13 |     Matrix () {
14 |         rows = 0;
15 |         cols = 0;
16 |         a = new double*[rows];
17 |         for(int i = 0; i < rows; i++)
18 |             a[i] = new double[cols];
19 |     }
20 |
21 |     Matrix(int n, int m, bool identity = 0)
22 |     {
23 |         rows = n;
24 |         cols = m;
25 |         a = new double *[rows];
26 |         for (int i = 0; i < rows; i++){
27 |             a[i] = new double[cols];
28 |             for (int j = 0; j < cols; j++){
29 |                 a[i][j] = identity * (i == j);
30 |             }
31 |         }
32 |     }
33 |
34 |     double* operator[] (int i)
35 |     {
36 |         return a[i];
37 |     }
38 |
39 |     void RowSwap(int row1, int row2)
40 |     {
41 |         swap(a[row1], a[row2]);
42 |     }
43 |
44 |     void scan()
45 |     {
46 |         for(int i = 0; i < rows; i++)
47 |         {
```

```

48         for(int j = 0; j < cols; j++)
49         {
50             scanf ("%lf", &a[i][j]);
51         }
52     }
53 }
54
55 void print()
56 {
57     for(int i = 0; i < rows; i++)
58     {
59         for(int j = 0; j < cols; j++)
60         {
61             cout <<"\t"<< a[i][j] << "\t";
62         }
63         cout << endl;
64     }
65 }
66
67 Matrix& operator*=(Matrix& m)
68 {
69     if (cols != m.rows)
70     {
71         throw "Wait. That's illegal.";
72     }
73     Matrix temp(m.rows, m.cols);
74     for (int i = 0; i < temp.rows; ++i)
75     {
76         for (int j = 0; j < temp.cols; ++j)
77         {
78             for (int k = 0; k < cols; ++k)
79             {
80                 temp.a[i][j] += (a[i][k] * m.a[k][j]);
81             }
82         }
83     }
84     return (*this = temp);
85 }
86 };
87
88
89 Matrix operator* (Matrix & A, double k){
90     Matrix C = Matrix (A.rows, A.cols);
91     for (int i = 0; i < A.rows; ++ i){
92         for (int j = 0; j < A.cols; ++ j){
93             C[i][j] = A[i][j] * k;
94         }
95     }
96     return C;

```

```

97 }
98
99 Matrix operator* (Matrix & A, Matrix & B){
100     Matrix C = Matrix (A.rows, B.cols);
101     for (int i = 0; i < A.rows; ++ i){
102         for (int j = 0; j < B.cols; ++ j){
103             for (int k = 0; k < A.cols; ++ k){
104                 C[i][j] += A[i][k] * B[k][j];
105             }
106         }
107     }
108     return C;
109 }
110
111
112 double* LU(Matrix U, Matrix B){
113     int n = U.rows;
114
115     Matrix M[n - 1];
116     Matrix L = Matrix(n, n, 1);
117     int p = 0;
118
119     for (int k = 0; k < n - 1; ++k){
120         M[k] = Matrix(n, n, 1);
121         for (int i = k + 1; i < n; ++i){
122             if (U[k][k] == 0){
123                 int j = k + 1;
124                 while (U[j][j] == 0 and j < n){
125                     j += 1;
126                 }
127                 if (j == n){
128                     break;
129                 }
130                 U.RowSwap(k, j);
131                 B.RowSwap(k, j);
132                 p += 1;
133             }
134             M[k][i][k] = U[i][k] / U[k][k];
135             for (int j = k; j < n; ++j){
136                 U[i][j] -= M[k][i][k] * U[k][j];
137             }
138         }
139         L *= M[k];
140     }
141
142     double Z[n];
143     for (int i = 0; i < n; ++i){
144         double s = 0;
145         for (int j = 0; j < i; ++j){

```

```

146         s += L[i][j] * Z[j];
147     }
148     Z[i] = B[i][0] - s;
149 }
150
151 double* X = new double[n];
152 for (int i = n - 1; i > -1; --i){
153     double s = 0;
154     for (int j = i + 1; j < n; ++j){
155         s += U[i][j] * X[j];
156     }
157     X[i] = (Z[i] - s) / U[i][i];
158 }
159 return X;
160 }
161
162 double* SumN(double X[], int n, int k){
163     double* S = new double[k];
164     for (int j = 0; j < k; ++j){
165         double s = 0;
166         for (int i = 0; i < n; ++i){
167             s += pow(X[i], j);
168         }
169         S[j] = s;
170     }
171     return S;
172 }
173
174 double* LeastSQ(double X[], double Y[], int n, int k){
175     Matrix A = Matrix(k + 1, k + 1);
176     Matrix B = Matrix(k + 1, 1);
177     double* S = SumN(X, n, 2 * k + 1);
178     for (int i = 0; i < k + 1; ++i){
179         for (int j = i; j < k + 1; ++j){
180             A[i][j] = S[i + j];
181             A[j][i] = S[i + j];
182         }
183         double s = 0;
184         for (int j = 0; j < n; ++j){
185             s += Y[j] * pow(X[j], i);
186         }
187         B[i][0] = s;
188     }
189     return LU(A, B);
190 }
191
192 double* LeastSQF(double a[], double X[], int n, int k){
193     double* S = new double[n];
194     for (int i = 0; i < n; ++i){

```

```

195     double s = 0;
196     for (int j = 0; j < k + 1; ++j){
197         s += a[j] * pow(X[i], j);
198     }
199     S[i] = s;
200 }
201 return S;
202 }
203
204 double ErrSQ(double F[], double Y[], int n){
205     double s = 0;
206     for (int i = 0; i < n; ++i){
207         s += pow(F[i] - Y[i], 2);
208     }
209     return s;
210 }
211
212 int main()
213 {
214     int n = 6;
215     double X[n] = {-1.0, 0.0, 1.0, 2.0, 3.0, 4.0};
216     double Y[n] = {-0.5, 0.0, 0.5, 0.86603, 1.0, 0.86603};
217
218     int k = 3;
219     for (int i = 1; i <= k; ++i){
220         cout << ErrSQ(LeastSQF(LeastSQ(X, Y, n, i), X, n, i), Y, n) << " ";
221     }
222 }

```

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: 1

1. $X^* = 1.0$

i	0	1	2	3	4
x_i	-1.0	0.0	1.0	2.0	3.0
y_i	-0.5	0.0	0.50	0.86603	1.0

Рис. 6: Условия

11 Результаты работы



```
0.48206 -0.23206
```

Рис. 7: Вывод программы в консоли

12 Исходный код

```
1  #include <iostream>
2  #include <vector>
3  #include <string>
4  #include <cmath>
5
6  using namespace std;
7
8  int main(){
9      int n = 5;
10     double X[n] = {-1.0, 0.0, 1.0, 2.0, 3.0};
11     double Y[n] = {-0.5, 0.0, 0.5, 0.86603, 1.0};
12     double x = 1.0;
13
14     int i;
15     for (int j = 0; j < n - 1; ++j){
16         if (X[j] <= x and x <= X[j + 1]){
17             i = j;
18         }
19     }
20
21     //
22     double dy = (Y[i + 1] - Y[i]) / (X[i + 1] - X[i]) + ((Y[i + 2] - Y[i + 1]) / (X[i + 2] - X[i + 1]) - (Y[i + 1] - Y[i]) / (X[i + 1] - X[i])) / (X[i + 2] - X[i]) * (2 * x - X[i] - X[i + 1]);
23
24     //
25     double ddy = 2 * ((Y[i + 2] - Y[i + 1]) / (X[i + 2] - X[i + 1]) - (Y[i + 1] - Y[i]) / (X[i + 1] - X[i])) / (X[i + 2] - X[i]);
26     cout << dy << " " << ddy;
27 }
```


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: 1

$$y = x/(2x + 5)$$

$$X_0 = -1, X_k = 1, h_1 = 0.5, h_2 = 0.25$$

14 Результаты работы

```
-0.0545011 -0.057948
-0.0684524 -0.0614767
-0.0595238 -0.0591515
-0.0544465 -0.0591515 -0.0590274
```

Рис. 8: Вывод программы в консоли

15 Исходный код

```
1 | #include <iostream>
2 | #include <vector>
3 | #include <string>
4 | #include <cmath>
5 |
6 | using namespace std;
7 |
8 | double func(double x){
9 |     return x / (2 * x + 5);
10 | }
11 |
12 | double Rect(double a, double b, double h){
13 |     int n = (b - a) / h;
14 |     double X[n + 1];
15 |     for (int i = 0; i <= n; ++i){
16 |         X[i] = a + h * i;
17 |     }
18 |
19 |     double F = 0;
20 |     for (int i = 0; i < n; ++i){
21 |         F += func((X[i] + X[i + 1]) / 2);
22 |     }
23 |     return F * h;
24 | }
25 |
26 | double Trap(double a, double b, double h){
27 |     int n = (b - a) / h;
28 |     double X[n + 1];
29 |     for (int i = 0; i <= n; ++i){
30 |         X[i] = a + h * i;
31 |     }
32 |
33 |     double F = 0;
34 |     for (int i = 0; i < n; ++i){
35 |         F += func(X[i]) + func(X[i + 1]);
36 |     }
37 |     return F * h / 2;
38 | }
39 |
40 | double S(double a, double b, double h){
41 |     int n = (b - a) / 2 / h;
42 |     double X[n + 1];
43 |     for (int i = 0; i <= n; ++i){
44 |         X[i] = a + 2 * h * i;
45 |     }
46 |
47 |     double F = 0;
```

```

48     for (int i = 0; i < n; ++i){
49         F += func(X[i]) + 4 * func((X[i] + X[i + 1]) / 2) + func(X[i + 1]);
50     }
51     return F * h / 3;
52 }
53
54 double RR(double F1, double F2, double h1, double h2, int p){
55     return F1 + (F1 - F2) / (pow(h2 / h1, p) - 1);
56 }
57
58 int main(){
59     double xo = -1;
60     double xk = 1;
61     double h1 = 0.5;
62     double h2 = 0.25;
63
64     cout << Rect(xo, xk, h1) << " " << Rect(xo, xk, h2) << "\n";
65     cout << Trap(xo, xk, h1) << " " << Trap(xo, xk, h2) << "\n";
66     cout << S(xo, xk, h1) << " " << S(xo, xk, h2) << "\n";
67     cout << "\n";
68     cout << RR(Trap(xo, xk, h1), Rect(xo, xk, h2), h1, h2, 2) << " " << RR(Trap(xo, xk,
        h1), Trap(xo, xk, h2), h1, h2, 2) << " " << RR(S(xo, xk, h1), S(xo, xk, h2),
        h1, h2, 2) << "\n";
69 }

```