

**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: Тысячный В.В.
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: 25

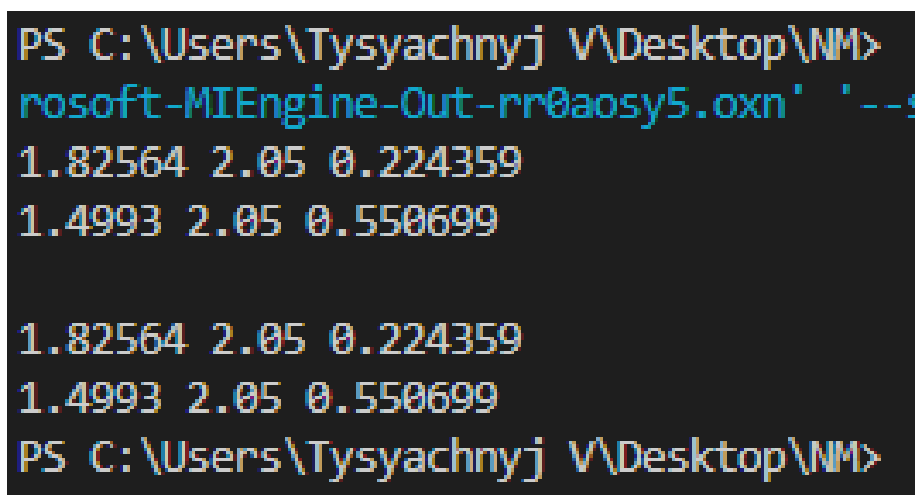
$$y = 1/x + x,$$

a) $X_i = 0.1, 0.5, 0.9, 1.3;$

b) $X_i = 0.1, 0.5, 1.1, 1.3;$

$$X^* = 0.8$$

2 Результаты работы



```
PS C:\Users\Tisyachnyj V\Desktop\NM>
rossoft-MIEngine-Out-rr0aosy5.oxn' '--s
1.82564 2.05 0.224359
1.4993 2.05 0.550699

1.82564 2.05 0.224359
1.4993 2.05 0.550699
PS C:\Users\Tisyachnyj V\Desktop\NM>
```

Рис. 1: Вывод программы в консоли

3 Исходный код

```
1 | #include <iostream>
2 | #include <vector>
3 | #include <string>
4 | #include <cmath>
5 |
6 | using namespace std;
7 |
8 | //
9 | double f(double x){
10 |     return 1 / x + x;
11 | }
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 | //      n
29 | double Split_difference(double X[], double Y[], int n){
30 |     if (n == 0){
31 |         return Y[0];
32 |     }
33 |     else if (n == 1){
34 |         return (Y[0] - Y[1]) / (X[0] - X[1]);
35 |     }
36 |     else{
37 |         double X_a[n];
38 |         double X_b[n];
39 |         double Y_a[n];
40 |         double Y_b[n];
41 |         for (int i = 0; i < n; ++i){
42 |             X_a[i] = X[i];
43 |             X_b[i] = X[i + 1];
44 |             Y_a[i] = Y[i];
45 |             Y_b[i] = Y[i + 1];
46 |         }
```

```

47         return (Split_difference(X_a, Y_a, n - 1) - Split_difference(X_b, Y_b, n - 1))
48             / (X[0] - X[n]);
49     }
50 }
51 //
52 double Newton(double X[], double Y[], int n, double x){
53     double s = 0;
54     double X_n[n];
55     double Y_n[n];
56     for (int i = 0; i < n; ++i){
57         X_n[i] = X[i];
58         Y_n[i] = Y[i];
59         double p = Split_difference(X_n, Y_n, i);
60         for (int j = 0; j < i; ++j){
61             p *= (x - X[j]);
62         }
63         s += p;
64     }
65     return s;
66 }
67
68
69 int main(){
70     int n = 4;
71     double X_1[n] = {0.1, 0.5, 0.9, 1.3};
72     double X_2[n] = {0.1, 0.5, 1.1, 1.3};
73     double Y_1[n] = {f(X_1[0]), f(X_1[1]), f(X_1[2]), f(X_1[3])};
74     double Y_2[n] = {f(X_2[0]), f(X_2[1]), f(X_2[2]), f(X_2[3])};
75     double x = 0.8;
76
77     //
78     double L_1 = Lagrange(X_1, Y_1, n, x);
79     cout << L_1 << " " << f(x) << " " << abs(L_1 - f(x)) << "\n";
80     double L_2 = Lagrange(X_2, Y_2, n, x);
81     cout << L_2 << " " << f(x) << " " << abs(L_2 - f(x)) << "\n";
82
83     cout << "\n";
84
85     //
86     double N_1 = Newton(X_1, Y_1, n, x);
87     cout << N_1 << " " << f(x) << " " << abs(N_1 - f(x)) << "\n";
88     double N_2 = Newton(X_2, Y_2, n, x);
89     cout << N_2 << " " << f(x) << " " << abs(N_2 - f(x)) << "\n";
90 }

```

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: 25

25. $X^* = 0.8$

i	0	1	2	3	4
x_i	0.1	0.5	0.9	1.3	1.7
f_i	10.1	2.5	2.0111	2.0692	2.2882

Рис. 2: Условия

5 Результаты работы

```
PS C:\Users\Tsyachnyj V\Desktop\NM>
rossoft-MIEngine-Out-0jff4alz.nqg' '--
1.80512
PS C:\Users\Tsyachnyj V\Desktop\NM>
```

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  {
10 private:
11     double **a;
12     int n, m;
13 public:
14     //
15     matrix (){
16         a = 0;
17         n = 0;
18         m = 0;
19     }
20
21     // NxM, E, ,
22     matrix (int N, int M, bool E = 0){
23         n = N;
24         m = M;
25         a = new double *[n];
26         for (int i = 0; i < n; ++ i){
27             a[i] = new double[m];
28             for (int j = 0; j < m; ++ j){
29                 a[i][j] = (i == j) * E;
30             }
31         }
32     }
33
34     double* operator [] (int index){
35         if (index >= 0 && index < n){
36             return a[index];
37         }
38         return 0;
39     }
40 };
41
42 double spline(double X[], double Y[], int n, double x){
43     double h[n];
44     for (int i = 0; i < n; ++i){
45         h[i] = X[i + 1] - X[i];
46     }
47 }
```

```

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

```

```

97 |     while (X[i] < x and X[i + 1] < x){
98 |         i += 1;
99 |     }
100 |     return a[i] + b[i] * (x - X[i]) + c[i] * pow(x - X[i], 2) + d[i] * pow(x - X[i], 3)
    |         ;
101 | }
102 |
103 | int main()
104 | {
105 |     int n = 4;
106 |     double X[n + 1] = {0.1, 0.5, 0.9, 1.3, 1.7};
107 |     double Y[n + 1] = {10.1, 2.5, 2.0111, 2.0692, 2.2882};
108 |     double x = 0.8;
109 |
110 |     cout << spline(X, Y, n, x);
111 | }

```


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: 25

25.

i	0	1	2	3	4	5
x_i	0.1	0.5	0.9	1.3	1.7	2.1
y_i	10.1	2.5	2.0111	2.0692	2.2882	2.5762

Рис. 4: Условия

8 Результаты работы

```
PS C:\Users\Tysyachnyj V\Desktop\NM>
rossoft-MIEngine-Out-xmdzbwls.weh' '--s
30.2546 8.98172 1.63329
PS C:\Users\Tysyachnyj V\Desktop\NM>
```

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  {
10 private:
11     double **a;
12     int n, m;
13 public:
14     //
15     matrix (){
16         a = 0;
17         n = 0;
18         m = 0;
19     }
20
21     // NxM, E, ,
22     matrix (int N, int M, bool E = 0){
23         n = N;
24         m = M;
25         a = new double *[n];
26         for (int i = 0; i < n; ++ i){
27             a[i] = new double[m];
28             for (int j = 0; j < m; ++ j){
29                 a[i][j] = (i == j) * E;
30             }
31         }
32     }
33
34     //
35     int get_n_rows(){
36         return n;
37     }
38     int get_n_cols(){
39         return m;
40     }
41
42     double* operator [] (int index){
43         return getRow (index);
44     }
45
46
47     //
```

```

48 double* getRow(int index){
49     if (index >= 0 && index < n){
50         return a[index];
51     }
52     return 0;
53 }
54
55 //
56 double* getColumn(int index){
57     if (index < 0 || index >= m){
58         return 0;
59     }
60     double * c = new double [n];
61     for (int i = 0; i < n; ++ i){
62         c[i] = a[i][index];
63     }
64     return c;
65 }
66
67 //
68 void swapRows (int index1, int index2){
69     if (index1 < 0 || index2 < 0 || index1 >= n || index2 >= n){
70         return ;
71     }
72     for (int i = 0; i < m; ++ i){
73         swap (a[index1][i], a[index2][i]);
74     }
75 }
76 };
77
78 //
79 matrix scanMatrix(int n, int m){
80     matrix a = matrix (n, m);
81     for (int i = 0; i < n; ++ i){
82         for (int j = 0; j < m; ++ j){
83             scanf ("%lf", & a[i][j]);
84         }
85     }
86     return a;
87 }
88
89 //
90 void printMatrix (matrix & a){
91     for (int i = 0; i < a.get_n_rows (); ++ i){
92         for (int j = 0; j < a.get_n_cols (); ++ j){
93             printf ("%5.3lf ", a[i][j]);
94         }
95         puts ("");
96     }

```

```

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

```

```

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

```

```

195     matrix A = matrix(k + 1, k + 1);
196     matrix B = matrix(k + 1, 1);
197     double* S = sum_n(X, n, 2 * k + 1);
198     for (int i = 0; i < k + 1; ++i){
199         for (int j = i; j < k + 1; ++j){
200             A[i][j] = S[i + j];
201             A[j][i] = S[i + j];
202         }
203         double s = 0;
204         for (int j = 0; j < n; ++j){
205             s += Y[j] * pow(X[j], i);
206         }
207         B[i][0] = s;
208     }
209     return LU(A, B);
210 }
211
212 //
213 double* F_MLS(double a[], double X[], int n, int k){
214     double* S = new double[n];
215     for (int i = 0; i < n; ++i){
216         double s = 0;
217         for (int j = 0; j < k + 1; ++j){
218             s += a[j] * pow(X[i], j);
219         }
220         S[i] = s;
221     }
222     return S;
223 }
224
225 //
226 double SE(double F[], double Y[], int n){
227     double s = 0;
228     for (int i = 0; i < n; ++i){
229         s += pow(F[i] - Y[i], 2);
230     }
231     return s;
232 }
233
234 int main()
235 {
236     int n = 6;
237     double X[n] = {0.1, 0.5, 0.9, 1.3, 1.7, 2.1};
238     double Y[n] = {10.1, 2.5, 2.011, 2.0692, 2.2882, 2.5762};
239
240     int k = 3;
241     for (int i = 1; i <= k; ++i){
242         cout << SE(F_MLS(MLS(X, Y, n, i), X, n, i), Y, n) << " ";
243     }

```


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: 25

25. $X^* = 2.0$

i	0	1	2	3	4
x_i	0.0	1.0	2.0	3.0	4.0
y_i	0.0	0.5	1.7321	3.0	3.4641

Рис. 6: Условия

11 Результаты работы

```
PS C:\Users\Tsyachnyj V\Desktop\NM>  
rosoft-MIEngine-Out-jyhplzmf.o1b' '--s  
1.6698 -0.8038  
PS C:\Users\Tsyachnyj V\Desktop\NM>
```

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] = {0., 1., 2., 3., 4.};
11     double Y[n] = {0.0, 0.5, 1.7321, 3.0, 3.4641};
12     double x = 2.0;
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

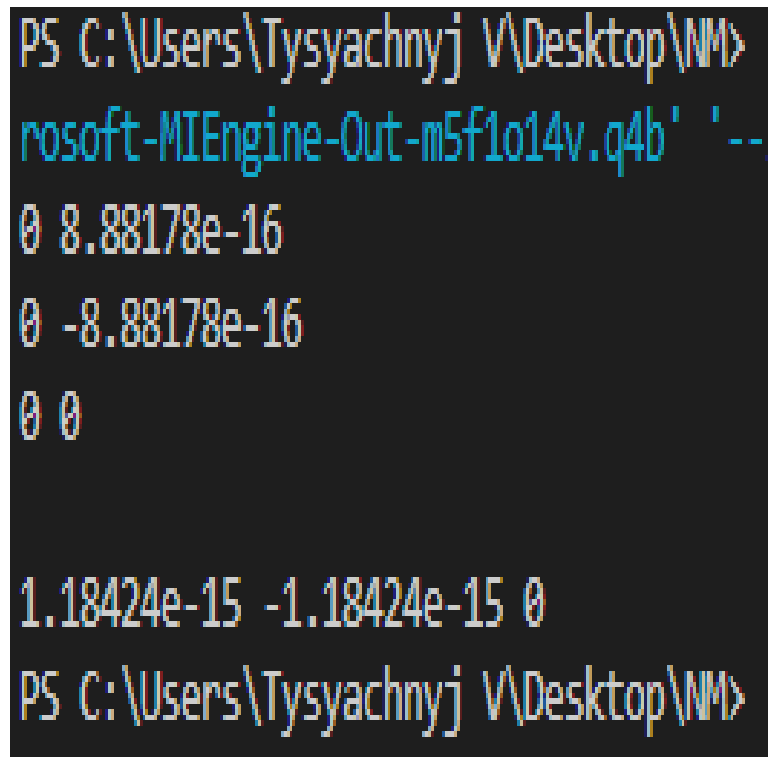
Вычислить определенный интеграл $\int_{X_0}^{X_1} y dx$, методами прямоугольников, трапеций, Симпсона с шагами h_1, h_2 . Оценить погрешность вычислений, используя Метод Рунге-Ромберга:

Вариант: 25

$$y = x\sqrt{49 - x^2}$$

$$X_0 = -2, X_k = 2, h_1 = 1.0, h_2 = 0.5$$

14 Результаты работы



```
PS C:\Users\Tysyachnyj V\Desktop\NM>
rosoft-MIEngine-Out-m5f1o14v.q4b' '--
0 8.88178e-16
0 -8.88178e-16
0 0

1.18424e-15 -1.18424e-15 0
PS C:\Users\Tysyachnyj V\Desktop\NM>
```

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 f(double x){
9 |     return x * pow(49 - pow(x, 2), 0.5);
10 | }
11 |
12 | //
13 | double rectangle(double a, double b, double h){
14 |     int n = (b - a) / h;
15 |     double X[n + 1];
16 |     for (int i = 0; i <= n; ++i){
17 |         X[i] = a + h * i;
18 |     }
19 |
20 |     double F = 0;
21 |     for (int i = 0; i < n; ++i){
22 |         F += f((X[i] + X[i + 1]) / 2);
23 |     }
24 |     return F * h;
25 | }
26 |
27 | //
28 | double trapezoid(double a, double b, double h){
29 |     int n = (b - a) / h;
30 |     double X[n + 1];
31 |     for (int i = 0; i <= n; ++i){
32 |         X[i] = a + h * i;
33 |     }
34 |
35 |     double F = 0;
36 |     for (int i = 0; i < n; ++i){
37 |         F += f(X[i]) + f(X[i + 1]);
38 |     }
39 |     return F * h / 2;
40 | }
41 |
42 | //
43 | double Simpson(double a, double b, double h){
44 |     int n = (b - a) / 2 / h;
45 |     double X[n + 1];
46 |     for (int i = 0; i <= n; ++i){
47 |         X[i] = a + 2 * h * i;
```

```

48     }
49
50     double F = 0;
51     for (int i = 0; i < n; ++i){
52         F += f(X[i]) + 4 * f((X[i] + X[i + 1]) / 2) + f(X[i + 1]);
53     }
54     return F * h / 3;
55 }
56
57 // -
58 double Runge_Romberg(double F1, double F2, double h1, double h2, int p){
59     return F1 + (F1 - F2) / (pow(h2 / h1, p) - 1);
60 }
61
62 int main(){
63     double a = -2;
64     double b = 2;
65     double h1 = 1;
66     double h2 = 0.5;
67
68     cout << rectangle(a, b, h1) << " " << rectangle(a, b, h2) << "\n";
69     cout << trapezoid(a, b, h1) << " " << trapezoid(a, b, h2) << "\n";
70     cout << Simpson(a, b, h1) << " " << Simpson(a, b, h2) << "\n";
71     cout << "\n";
72     cout << Runge_Romberg(rectangle(a, b, h1), rectangle(a, b, h2), h1, h2, 2) << " "
73         << Runge_Romberg(trapezoid(a, b, h1), trapezoid(a, b, h2), h1, h2, 2) << " " <<
74         Runge_Romberg(Simpson(a, b, h1), Simpson(a, b, h2), h1, h2, 2) << "\n";
75 }

```