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: Ю. В. Кон
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 $X_i, i = 0, ..3$ построить интерполяционные многочлены Лагранжа и Ньютона, проходящие через точки $\{X_i, Y_i\}$. Вычислить значение погрешности интерполяции в точке X^*.

Вариант: 12

$y = sin(x) + x, a) X_i = 0, \pi/6, 2\pi/6, 3\pi/6;) X_i = 0, \pi/6, \pi/4, \pi/2; X^* = 1.0$

2 Результаты работы

```
Коэффициенты:
-0
3.56536
-6.66407
2.98484
Значение многочлена Лагранжа: 1.84109
Погрешность интерполяции в точке X: 0.000384968
Коэффициенты:
0
1.95493
-0.208608
-0.121411
Значение многочлена Ньютона: 1.84314
Погрешность интерполяции в точке X: 0.00166512
```

Рис. 1: Вывод программы в консоли

3 Исходный код

Файл с первым заданием третьей лабораторной работы:

```
#include <iostream>
#include <cmath>
#include <vector>
#include <fstream>

using namespace std;

double f(double x) {
    return (sin(x)+x);
}

pair<vector<double>, double> lagrange(vector<double> x, vector<double> y, double X,
    int n) {
    double res = 0.0;
    vector<double> coeff(n, 0);
    for (int i = 0; i < n; ++i) {
        coeff[i] = y[i];
        double l = y[i];
        for (int j = 0; j < n; ++j) {
            if (j != i) {
                coeff[i] = coeff[i]/(x[i] - x[j]);
                l *= (X - x[j]) / (x[i] - x[j]);
            }
        }
        res += l;
    }
    return make_pair(coeff, res);
}

pair<vector<double>, double> newton(vector<double> x, vector<double> y, double X, int
    n) {
    double res = 0.0;
    vector<double> coeff(n, 0);
    vector<vector<double>> f(n, vector<double>(n, 0));
    for (int i = 0; i < n; ++i) {
        f[i][0] = y[i];
    }
    for (int i = 1; i < n; ++i) {
        for (int j = 0; j < n - i; ++j) {
            f[j][i] = (f[j + 1][i - 1] - f[j][i - 1]) / (x[i + j] - x[j]);
        }
    }
    for (int i = 0; i < n; ++i) {
        coeff[i] = f[0][i];
        double tmp = f[0][i];
        for (int j = 0; j < i; ++j) {
```

```
45              tmp *= (X - x[j]);
46          }
47          res += tmp;
48      }
49      return make_pair(coeff, res);
50  }
51
52  int main(){
53      int n=4;
54      vector<double> x_a = {0.0, M_PI/6, 2* M_PI/6, 3* M_PI/6};
55      vector<double> y_a;
56      vector<double> coeff_a(n, 0);
57      double X = 1.0, res, delta;
58      for (double xi : x_a) {
59          y_a.push_back(f(xi));
60      }
61      coeff_a = lagrange(x_a, y_a, X, n).first;
62      res = lagrange(x_a, y_a, X, n).second;
63
64      ofstream fout("output.txt");
65
66      fout << ":" << endl;
67      for (int i = 0; i < n; ++i) {
68          fout << coeff_a[i] << endl;
69      }
70
71      fout << "  : " << res << endl;
72      delta = abs(f(X)-res);
73      fout << "    X: " << delta << endl;
74
75      vector<double> x_b = {0.0, M_PI/6, M_PI/4, M_PI/2};
76      vector<double> y_b;
77      vector<double> coeff_b(n, 0);
78      res = 0.0;
79      delta = 0.0;
80      for (double xi : x_b) {
81          y_b.push_back(f(xi));
82      }
83      coeff_b = newton(x_b, y_b, X, n).first;
84      res = newton(x_b, y_b, X, n).second;
85      fout << ":" << endl;
86      for (int i = 0; i < n; ++i) {
87          fout << coeff_b[i] << endl;
88      }
89
90      fout << "  : " << res << endl;
91      delta = abs(f(X)-res);
92      fout << "    X: " << delta << endl;
93      return 0;
```

94 || }

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: 12

$X_i = 0.8$

i	0	1	2	3	4
x_i	0.0	0.5	1.0	1.5	2.0
f_i	0.0	0.97943	1.8415	2.4975	2.9093

Рис. 2: Условие

5 Результаты работы

```
Коэффициенты:
0 2.00101 0 -0.168617
0.97943 1.87455 -0.252926 -0.0957943
1.8415 1.54978 -0.396617 -0.157886
2.4975 1.03475 -0.633446 0.422297
Значение функции в точке X: 1.51645
```

Рис. 3: Вывод программы в консоли

6 Исходный код

Файл со вторым заданием третьей лабораторной работы:

```
#include <iostream>
#include <cmath>
#include <vector>
#include <fstream>

using namespace std;

double spline(vector<double> x, vector<double> y, double X, int n, vector<double>& a,
    vector<double>& b, vector<double>& c, vector<double>& d) {

    vector<double> h(n - 1);
    vector<double> p(n - 1);
    vector<double> l(n);
    vector<double> mu(n - 1);
    vector<double> z(n);

    for (int i = 0; i < n - 1; ++i) {
        h[i] = x[i + 1] - x[i];
    }

    for (int i = 1; i < n - 1; ++i) {
        p[i] = 3 * (y[i + 1] - y[i]) / h[i] - 3 * (y[i] - y[i - 1]) / h[i - 1];
    }

    l[0] = 1;
    mu[0] = 0;
    z[0] = 0;

    for (int i = 1; i < n - 1; ++i) {
        l[i] = 2 * (x[i + 1] - x[i - 1]) - h[i - 1] * mu[i - 1];
        mu[i] = h[i] / l[i];
        z[i] = (p[i] - h[i - 1] * z[i - 1]) / l[i];
    }

    l[n - 1] = 1;
    z[n - 1] = 0;

    for (int i = 0; i < n - 1; ++i){
        a[i] = y[i];
    }

    for (int j = n - 2; j >= 0; --j) {
        c[j] = z[j] - mu[j] * c[j + 1];
        b[j] = (y[j + 1] - y[j]) / h[j] - h[j] * (c[j + 1] + 2 * c[j]) / 3;
        d[j] = (c[j + 1] - c[j]) / (3 * h[j]);
    }
```

```
46
47      int index = 0;
48      for (int i = 0; i < n - 1; ++i) {
49          if (x[i] <= X && X <= x[i + 1]) {
50              index = i;
51              break;
52          }
53      }
54
55      double res = a[index] + b[index] * (X - x[index]) + c[index] * pow((X- x[index]),
            2) + d[index] * pow((X - x[index]), 3);
56
57      return res;
58  }
59
60  int main() {
61      int n = 5;
62      vector<double> x = {0.0, 0.5, 1.0, 1.5, 2.0};
63      double X = 0.8;
64      vector<double> y = {0.0, 0.97943, 1.8415, 2.4975, 2.9093};
65      vector<double> a(n-1, 0), b(n-1, 0), c(n-1, 0), d(n-1, 0);
66
67      ofstream fout("output.txt");
68
69      double res = spline(x, y, X, n, a, b, c, d);
70
71      fout << ":" << endl;
72      for (int i = 0; i < n-1; ++i) {
73          fout << a[i] << " " << b[i] << " " << c[i] << " " << d[i] << endl;
74      }
75
76      fout << "    X: " << res << endl;
77
78      return 0;
79  }
```

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: 12

i	0	1	2	3	4	5
x_i	-1.0	0.0	1.0	2.0	3.0	4.0
y_i	-1.8415	0.0	1.8415	2.9093	3.1411	3.2432

Рис. 4: Условия

8 Результаты работы

```
Коэффициенты приближающего многочлена первой степени:
0.00973619 1.02613
Сумма квадратов ошибок: 2.80941
Коэффициенты приближающего многочлена второй степени:
0.00973619 1.02613 3.93464e-270
Сумма квадратов ошибок: 0.0821187
```

Рис. 5: Вывод программы в консоли

9 Исходный код

Файл с третьим заданием третьей лабораторной работы:

```
#include <iostream>
#include <cmath>
#include <vector>
#include <fstream>

using namespace std;

void solve_system(vector<vector<double>>& a, vector<double>& b, vector<double>& x, int
     n) {
    for (int i = 0; i < n; ++i) {
        int pivot = i;
        for (int j = i + 1; j < n; ++j) {
            if (abs(a[j][i]) > abs(a[pivot][i])) {
                pivot = j;
            }
        }
        swap(a[i], a[pivot]);
        swap(b[i], b[pivot]);
        for (int j = i + 1; j < n; ++j) {
            double factor = a[j][i] / a[i][i];
            for (int k = i; k < n; ++k) {
                a[j][k] -= factor * a[i][k];
            }
            b[j] -= factor * b[i];
        }
    }
    x.assign(n, 0);
    for (int i = n - 1; i >= 0; --i) {
        double sum = 0;
        for (int j = i + 1; j < n; ++j) {
            sum += a[i][j] * x[j];
        }
        x[i] = (b[i] - sum) / a[i][i];
    }
}

vector<double> MNK(vector<double>& x, vector<double>& y, int degree, int n) {
    vector<vector<double>> a(degree + 1, vector<double>(degree + 1, 0));
    vector<double> b(degree + 1, 0);
    for (int i = 0; i <= degree; ++i) {
        for (int j = 0; j <= degree; ++j) {
            for (int k = 0; k < n; ++k) {
                a[i][j] += pow(x[k], i + j);
            }
        }
        for (int k = 0; k < n; ++k) {
```

```
46              b[i] += pow(x[k], i) * y[k];
47          }
48      }
49      vector<double> coefficients;
50      int n1 = a.size();
51      solve_system(a, b, coefficients, n1);
52      return coefficients;
53  }
54
55  double g(vector<double>& coefficients, double x) {
56      double result = 0;
57      for (size_t i = 0; i < coefficients.size(); ++i) {
58          result += coefficients[i] * pow(x, i);
59      }
60      return result;
61  }
62
63  double sum_of_errors(vector<double>& x, vector<double>& y, vector<double>&
        coefficients, int n) {
64      double sum = 0, error;
65      for (int i = 0; i < n; ++i) {
66          error = y[i] - g(coefficients, x[i]);
67          sum += error * error;
68      }
69      return sum;
70  }
71
72  int main(){
73      int n = 6, m = 1;
74      vector<double> x = {-1.0, 0.0, 1.0, 2.0, 3.0, 4.0};
75      vector<double> y = {-1.8415, 0.0, 1.8415, 2.9093, 3.1411, 3.2432};
76      ofstream fout("output.txt");
77      vector<double> coefficients_degree_1 = MNK(x, y, m, n);
78      double sum_degree_1 = sum_of_errors(x, y, coefficients_degree_1, n);
79      fout << "    :" << endl;
80      for (int i = 0; i < m+1; ++i) {
81          fout << coefficients_degree_1[i] << " ";
82      }
83      fout << endl;
84      fout << "  : " << sum_degree_1 << endl;
85
86      m = 2;
87      vector<double> coefficients_degree_2 = MNK(x, y, m, n);
88      double sum_degree_2 = sum_of_errors(x, y, coefficients_degree_2, n);
89      fout << "    :" << endl;
90      for (int i = 0; i < m+1; ++i) {
91          fout << coefficients_degree_1[i] << " ";
92      }
93      fout << endl;
```

```
94      fout << "  : " << sum_degree_2 << 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: 12 $X^* = 0.2$

i	0	1	2	3	4
x_i	-1.0	-0.4	0.2	0.6	1.0
y_i	-1.4142	-0.55838	0.27870	0.84008	1.4142

Рис. 6: Условия

11 Результаты работы

```
Первая производная:1.16538
Вторая производная:-0.765833
```

Рис. 7: Вывод программы в консоли

12 Исходный код

Файл с четвертым заданием третьей лабораторной работы:

```
#include <iostream>
#include <vector>
#include <fstream>

using namespace std;

pair<double, double> derivative(vector<double>& x, vector<double>& y, double X, int n)
     {
    int index = -1;
    for (int i = 0; i < n; ++i) {
        if (x[i] <= X)
            index = i;
        else
            break;
    }
    double h = x[1] - x[0];
    double derivative_1 = (y[index + 1] - y[index - 1]) / (2 * h);
    double derivative_2 = (y[index + 1] - 2 * y[index] + y[index - 1]) / (h * h);

    return make_pair(derivative_1, derivative_2);
}

int main(){
    int n = 5;
    vector<double> x = {-1.0, -0.4, 0.2, 0.6, 1.0};
    vector<double> y = {-1.4142, -0.55838, 0.27870, 0.84008, 1.4142};
    double X = 0.2;
    ofstream fout("output.txt");
    fout << " :" << derivative(x, y, X, n).first << endl;
    fout << " :" << derivative(x, y, X, n).second << endl;
    return 0;
}
```

3.5

13 Постановка задачи

Вычислить определенный интеграл $\int_{X_0}^{X_1} ydx$, методами прямоугольников, трапеций, Симпсона с шагами h_1, h_2. Оценить погрешность вычислений, используя Метод Рунге-Ромберга:
Вариант: 12
$y = x/(x^3 + 8)$ $X_0 = -1, X_k = 1, h_1 = 0.5, h_2 = 0.25$

14 Результаты работы

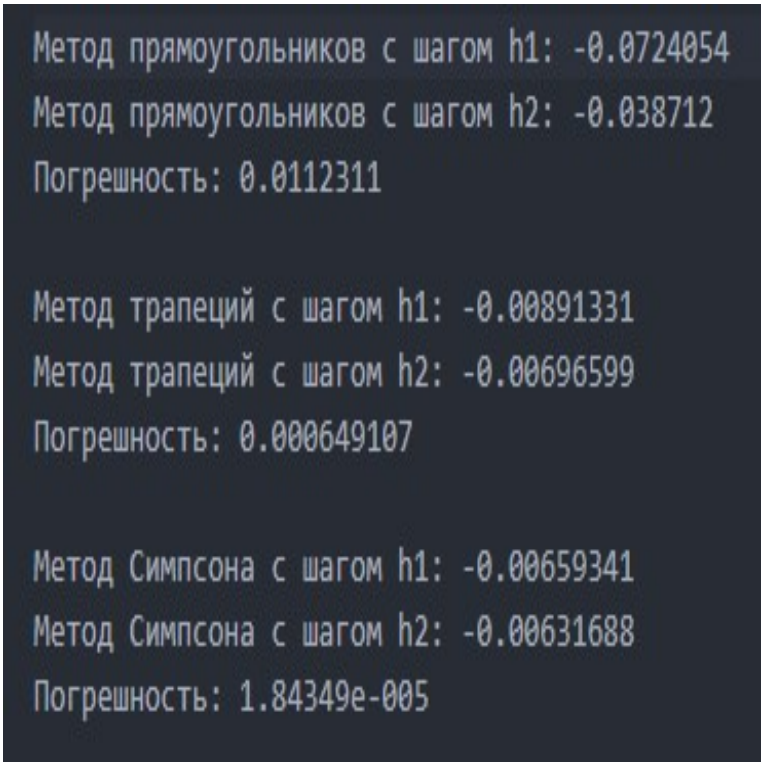

Рис. 8: Вывод программы в консоли

15 Исходный код

Файл с пятым заданием третьей лабораторной работы:

```cpp
#include <iostream>
#include <cmath>
#include <fstream>

using namespace std;

double y(double x) {
    return x / (pow(x, 3) + 8);
}

double rectangle(double x_0, double x_k, double h) {
    double res = 0.0;
    for (double x = x_0; x < x_k; x += h) {
        res += y(x) * h;
    }
    return res;
}

double trapezoid(double x_0, double x_k, double h) {
    double res = 0.0;
    for (double x = x_0; x < x_k; x += h) {
        res += (y(x) + y(x + h)) * h / 2.0;
    }
    return res;
}

double simpson(double x_0, double x_k, double h) {
    double res = 0.0;
    for (double x = x_0; x < x_k; x += 2 * h) {
        res += (y(x) + 4 * y(x + h) + y(x + 2 * h)) * h / 3.0;
    }
    return res;
}

double runge_romberg(double I1, double I2, double p) {
    return (I2 - I1) / (pow(2, p) - 1);
}

int main(){
    double x_0 = -1.0, x_k = 1.0, h1 = 0.5, h2 = 0.25;
    ofstream fout("output.txt");
    double rectangle_res_h1 = rectangle(x_0, x_k, h1);
    fout << "    h1: " << rectangle_res_h1 << endl;
    double rectangle_res_h2 = rectangle(x_0, x_k, h2);
    fout << "    h2: " << rectangle_res_h2 << endl;
    double delta_rectangle = runge_romberg(rectangle_res_h1, rectangle_res_h2, 2);
```

```
47      fout << ": " << delta_rectangle << endl;
48      fout << endl;
49
50      double trapezoid_res_h1 = trapezoid(x_0, x_k, h1);
51      fout << "    h1: " << trapezoid_res_h1 << endl;
52      double trapezoid_res_h2 = trapezoid(x_0, x_k, h2);
53      fout << "    h2: " << trapezoid_res_h2 << endl;
54      double delta_trapezoid = runge_romberg(trapezoid_res_h1, trapezoid_res_h2, 2);
55      fout << ": " << delta_trapezoid << endl;
56      fout << endl;
57
58      double simpson_res_h1 = simpson(x_0, x_k, h1);
59      fout << "    h1: " << simpson_res_h1 << endl;
60      double simpson_res_h2 = simpson(x_0, x_k, h2);
61      fout << "    h2: " << simpson_res_h2 << endl;
62      double delta_simpson = runge_romberg(simpson_res_h1, simpson_res_h2, 4);
63      fout << ": " << delta_simpson << endl;
64      return 0;
65  }
```