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

Вариант: 17

17. $y = e^x + x$, а) $X_i = -2, -1, 0, 1$; б) $X_i = -2, -1, 0.2, 1$; $X^* = -0.5$.

2 Результаты работы

```
Коэффициенты Лангранжа:
0.310777
-0.31606
-0.5
0.619714
Значение многочлена Лагранжа: 0.0910811
Погрешность интерполяции в точке x: 0.0154495
------------------------------------------------
Коэффициенты Ньютона:
-1.86466
1.23254
0.217602
0.120771
Значение многочлена Ньютона: 0.0839487
Погрешность интерполяции в точке x: 0.022582
```

Рис. 1: Вывод программы

3 Исходный код

```

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

using namespace std;

int main()
{
    int n = 5;
```

```
    ofstream fout;
    fout.open("output.txt");
    vector<vector<double>> A = {
            {-6, 5, 0, 0, 0},
            {-1, 13, 6, 0 ,0},
            {0, -9, -15, -4, 0},
            {0, 0, -1, -7, 1},
            {0, 0, 0, 9, -18}
    };
    vector<double> d = { 51, 100, -12, 47, -90 };
    vector<double> P(n, 0);
    vector<double> Q(n, 0);
    vector<double> x(n, 0);
    P[0] = -A[0][1] / A[0][0];
    Q[0] = d[0] / A[0][0];
    for (int i = 1; i < n; i++) {
        P[i] = -A[i][i + 1] / (A[i][i] + A[i][i - 1] * P[i - 1]);
        Q[i] = (d[i] - A[i][i - 1] * Q[i - 1]) / (A[i][i] + A[i][i - 1] * P[i - 1]);
    }
    for (int i = n - 1; i >= 0; i--) {
        x[i] = P[i] * x[i + 1] + Q[i];
    }
    for (int i = 0; i < n; i++) {
        fout << x[i] << endl;
    }
    return 0;
}
```

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

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

using namespace std;

vector<double> SolveSys(vector<vector<double>>& A, vector<double>& d, int n){
    vector<double> P (n, 0);
    vector<double> Q (n, 0);
```

```
    vector<double> x (n, 0);
    P[0] = - A[0][1] / A[0][0];
    Q[0] = d[0] / A[0][0];
    for (int i = 1; i < n; i++){
        P[i] = - A[i][i+1] / (A[i][i] + A[i][i-1] * P[i-1]);
        Q[i] = (d[i] - A[i][i-1] * Q[i-1]) / (A[i][i] + A[i][i-1] * P[i-1]);
    }
    for (int i = n - 1; i >= 0; i--){
        x[i] = P[i] * x[i+1] + Q[i];
    }
    return x;
}

double Spline(vector<double>& x, vector<double>& f, double x0, int n = 4){
    double res;
    vector<double> h (n+1,0);
    vector<double> a (n+1,0);
    vector<double> b (n+1,0);
    vector<double> c (n+1,0);
    vector<double> d (n+1,0);
    for (int i = 1; i <= n; i++){
        h[i] = x[i] - x[i-1];
    }
    vector<vector<double>> A = {
            {2*(h[1] + h[2]), h[2], 0},
            {h[2], 2*(h[2] + h[3]), h[3]},
            {0, h[3], 2*(h[3] + h[4])}
    };
    vector<double> B = {3*((f[2] - f[1])/h[2] - (f[1] - f[0])/h[1]),
                        3*((f[3] - f[2])/h[3] - (f[2] - f[1])/h[2]),
                        3*((f[4] - f[3])/h[4] - (f[3] - f[2])/h[3])};
    vector<double> c_0 = SolveSys(A, B, n-1);
    c[1] = 0;
    c[2] = c_0[0];
    c[3] = c_0[1];
    c[4] = c_0[2];
    for (int i = 1; i <= n; i++){
        a[i] = f[i-1];
    }
    for (int i = 1; i <= n-1; i++){
        b[i] = (f[i] - f[i-1])/h[i] - 1/3 * h[i]*(c[i+1] + 2*c[i]);
        d[i] = (c[i+1] - c[i])/(3*h[i]);
    }
    b[n] = (f[n] - f[n-1])/h[n] - 2/3*h[n]*c[n];
    d[n] = -c[n]/(3*h[n]);
    res = a[2] + b[2]*(x0 - x[1]) + c[2]*pow(x0 - x[1],2) + d[2]*pow(x0 - x[1],3);

    return res;
}
```

```

int main() {
    ofstream fout;
    fout.open("output.txt");
    vector<double> x = {-2, -1, 0, 1, 2};
    vector<double> f = {-1.8647, -0.63212, 1, 3.7183, 9.3891};
    double X0 = -0.5;
    fout << "    X: " << Spline(x, f, X0) << endl;
    return 0;
}
```

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

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

```

```
using namespace std;

vector<double> SolveSys(vector<vector<double>>& A, vector<double>& d, int n){
    vector<double> P (n, 0);
    vector<double> Q (n, 0);
    vector<double> x (n, 0);
    P[0] = - A[0][1] / A[0][0];
    Q[0] = d[0] / A[0][0];
    for (int i = 1; i < n; i++){
        P[i] = - A[i][i+1] / (A[i][i] + A[i][i-1] * P[i-1]);
        Q[i] = (d[i] - A[i][i-1] * Q[i-1]) / (A[i][i] + A[i][i-1] * P[i-1]);
    }
    for (int i = n - 1; i >= 0; i--){
        x[i] = P[i] * x[i+1] + Q[i];
    }
    return x;
}



tuple<vector<double>, vector<double>, double, double> MNK(vector<double>& x, vector<double>& y, int n){
    double xsum = 0, ysum = 0, xsum2 = 0, ysum2 = 0, xysum = 0, xsum3 = 0, xsum4 = 0, xysum2 = 0;

    for (int i = 0; i <= n; i++){
        xsum += x[i];
        ysum += y[i];
        xsum2 += x[i] * x[i];
        ysum2 += y[i] * y[i];
        xysum += x[i] * y[i];
        xsum3 += pow(x[i], 3);
        xsum4 += pow(x[i], 4);
        xysum2 += x[i] * y[i] * x[i];
    }
    vector<vector<double>> A1= {
            {(double)(n+1), xsum},
            {xsum, xsum2}
    };
    vector<double> B1 = {ysum, xysum};
    vector<double> coef1 = SolveSys(A1, B1, 2);
    vector<vector<double>> A2= {
            {(double)(n+1), xsum, xsum2},
            {xsum, xsum2, xsum3},
            {xsum2, xsum3, xsum4}
    };

    vector<double> B2 = {ysum, xysum, xysum2};
```

```
    vector<double> coef2 = SolveSys(A2, B2, 3);

    vector<double> f (n+1, 0);
    double err1 = 0;
    for (int i = 0; i <= n; i++){
        f[i] = coef1[0] + coef1[1] * x[i];
    }
    for (int i = 0; i <= n; i++){
        err1 += pow(f[i] - y[i], 2);
    }

    double err2 = 0;
    for (int i = 0; i <= n; i++){
        f[i] = coef2[0] + coef2[1] * x[i] + coef2[2] * x[i] * x[i];
    }
    for (int i = 0; i <= n; i++){
        err2 += pow(f[i] - y[i], 2);
    }

    return make_tuple(coef1, coef2, err1, err2);
}

int main() {
    ofstream fout;
    fout.open("output.txt");
    int n = 5;
    vector<double> x = {-3, -2, -1, 0, 1, 2};
    vector<double> y = {-2.9502, -1.8647, -0.63212, 1, 3.7183, 9.3891};
    auto[coefs1, coefs2, err1, err2] = MNK(x, y, n);
    fout << "  1 :" << endl;
    for (int i = 0; i < 2; i++){
        fout << coefs1[i] << " ";
    }
    fout << endl;
    fout << "C  : " << err1 << endl;
    fout << endl;
    fout << "  2 :" << endl;
    for (int i = 0; i < 3; i++){
        fout << coefs2[i] << " ";
    }
    fout << endl;
    fout << "C  : " << err2 << endl;
    return 0;
}
```

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

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

using namespace std;

double Diff1(vector<double>& x, vector<double>& y,double x0, int i){
    double res;
    res = (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*x0 - x[i] - x[i+1]);
    return res;
}

double Diff2(vector<double>& x, vector<double>& y,double x0, int i){
    double res;
```

```
    res = 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]));
    return res;
}

int main() {
    ofstream fout("output.txt");
    vector<double> x = {-0.2, 0, 0.2, 0.4, 0.6};
    vector<double> y = {-0.40136, 0.0, 0.40136, 0.81152, 1.2435};
    double x0 = 0.2;
    int p = 0;
    for (int i = 0; i < 5; i++){
        if (x0 >= x[i] && x0 <= x[i+1]){
            p = i;
            break;
        }
    }
    double res1 = Diff1(x, y, x0, p);
    double res2 = Diff2(x, y, x0, p);
    fout << "    : " << res1 << endl;
    fout << "    : " << res2 << endl;
    return 0;
}
```

3.5

13 Постановка задачи

Вычислить определенный интеграл $\int\limits_{X_0}^{X_1} ydx$, методами прямоугольников, трапеций, Симпсона с шагами h_1, h_2. Оценить погрешность вычислений, используя Метод Рунге-Ромберга:
Вариант: 17

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

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

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

using namespace std;

double Y(double x){
    return 1/(256 - pow(x,4));
}

pair <double, double> Rectangle(double x0, double xk, double h1, double h2){
    int n1 = 0, n2 = 0;
    double x = x0;
    double x1 = x0;
    double res = 0, res1 = 0;
    double h = 0;
    vector <double> xi1, xi2;
    while (x != xk){
        xi1.push_back(x);
        x += h1;
        n1++;
    }
    n1++;
    xi1.push_back(x);
    while (x1 != xk){
        xi2.push_back(x1);
        x1 += h2;
        n2++;
    }
    n2++;
    xi2.push_back(x1);

    for (int i = 1; i <= n1; i++){
        h = xi1[i] - xi1[i-1];
        res += h * Y((xi1[i] + xi1[i-1]) / 2);
    }

    for (int i = 1; i <= n2; i++){
        h = xi2[i] - xi2[i-1];
        res1 += h * Y((xi2[i] + xi2[i-1]) / 2);
    }
    return make_pair(res, res1);
}

pair <double, double> Trapez(double x0, double xk, double h1, double h2){
    int n1 = 0, n2 = 0;
```

```
    double x = x0;
    double x1 = x0;
    double res = 0, res1 = 0;
    double h = 0;
    vector <double> xi1, xi2;
    while (x != xk){
        xi1.push_back(x);
        x += h1;
        n1++;
    }
    n1++;
    xi1.push_back(x);
    while (x1 != xk){
        xi2.push_back(x1);
        x1 += h2;
        n2++;
    }
    n2++;
    xi2.push_back(x1);

    for (int i = 1; i <= n1; i++){
        h = xi1[i] - xi1[i-1];
        res += h * (Y(xi1[i]) + Y(xi1[i-1]));
    }
    res *= 0.5;
    for (int i = 1; i <= n2; i++){
        h = xi2[i] - xi2[i-1];
        res1 += h * (Y(xi2[i]) + Y(xi2[i-1]));
    }
    res1 *= 0.5;
    return make_pair(res, res1);
}

pair <double, double> Simpson(double x0, double xk, double h1, double h2){
    int n1 = 0, n2 = 0;
    double x = x0;
    double x1 = x0;
    double res = 0, res1 = 0;
    double h = 0;
    vector <double> xi1, xi2;
    while (x != xk){
        xi1.push_back(x);
        x += h1;
        n1++;
    }
    n1++;
    xi1.push_back(x);
    while (x1 != xk){
        xi2.push_back(x1);
```

```
        x1 += h2;
        n2++;
    }
    n2++;
    xi2.push_back(x1);

    for (int i = 1; i <= n1; i++){
        h = (xi1[i] - xi1[i-1]) / 2;
        res += h * (Y(xi1[i-1]) + 4 * Y((xi1[i-1] + xi1[i])/2) + Y(xi1[i]));
    }
    res = res / 3;
    for (int i = 1; i <= n2; i++){
        h = (xi2[i] - xi2[i-1]) / 2;
        res1 += h * (Y(xi2[i-1]) + 4 * Y((xi2[i-1] + xi2[i])/2) + Y(xi2[i]));;
    }
    res1 = res1 / 3;
    return make_pair(res, res1);
}

double Runge(double F1, double F2, double p){
    return F1 + (F1 - F2)/(pow(2, p) - 1);
}

int main() {
    ofstream fout("output.txt");
    double x0 = -2;
    double xk = 2;
    double h1 = 1;
    double h2 = 0.5;
    double res1 = Rectangle(x0, xk, h1, h2).first, res2 = Rectangle(x0, xk, h1, h2).second;
    fout << "----------   ----------" << endl;
    fout << "    " << h1 << ": " << res1 << endl << "    " << h2 << ": " << res2 << endl;
    fout << ": " << Runge(res1, res2, 2) << endl;
    fout << "----------   ----------" << endl;
    res1 = Trapez(x0, xk, h1, h2).first;
    res2 = Trapez(x0, xk, h1, h2).second;
    fout << "    " << h1 << ": " << res1 << endl << "    " << h2 << ": " << res2 << endl;
    fout << ": " << Runge(res1, res2, 2) << endl;
    fout << "----------   ----------" << endl;
    res1 = Simpson(x0, xk, h1, h2).first;
    res2 = Simpson(x0, xk, h1, h2).second;
    fout << "    " << h1 << ": " << res1 << endl << "    " << h2 << ": " << res2 << endl;
    fout << ": " << Runge(res1, res2, 2) << endl;
    return 0;
}
```