|  |  |
| --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ | |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  «Пермский государственный национальный исследовательский университет» | |
| **ЧИСЛЕННЫЕ МЕТОДЫ** *Лабораторная работа №\_5\_*  **«**Интерполирование. Среднеквадратичное приближение**»** | |
| Варианты: №3, №6 | |
|  | Работу выполнили студенты группы ПМИ-2-16  Мироненко Анастасия Олеговна,  Зимин Илья Владимирович |
| Оценка отчета   |  |  |  | | --- | --- | --- | |  | Баллы |  | | Конспект | 1 |  | | Опоздание с отчетом | -0.05 день |  | | Попытки | -0.5 попыт. |  | | Замечания к отчету | -0.25 зам. |  | | Программа | 1 |  | | Выводы  (Заключение) | 1 |  | | Защита | 1 |  | | ИТОГО: |  |  | | Проверил:  профессор, доктор физико-математических наук  С. В. Русаков  “\_\_\_\_” 2018 г. |
| Пермь 2018 | |

**Содержание**

[1. Задание. 3](#_Toc525161389)

[2. Исходные данные. 4](#_Toc525161390)

[3. Решение 5](#_Toc525161391)

[4. Краткие выводы. 6](#_Toc525161392)

[5. Текст программы. 7](#_Toc525161393)

1. **Задание**

*Задача* - приблизить заданную функцию  на отрезке .

1) Построить таблицу ,  .

По полученной таблице произвести интерполяцию с помощью

* Формулы Ньютона;
* Кубических сплайнов дефекта 1.

Провести оценку погрешности в узлах .

Сравнить оценку погрешности с реальной погрешностью.

2) Выполнить среднеквадратичное приближение заданной функции на заданном отрезке c помощью полинома второго порядка

* Дискретный вариант (по таблице из п.1);
* Непрерывный (интегральный) вариант.

Провести оценку погрешности.

Построить график приближаемой и приближающих функций.

3) Методом обратного и нтерполирования, используя интерполяционную формулу Ньютона, найти корень уравнения

.

Константа *с* выбирается таким образом, чтобы существовал корень на отрезке .

*Указание.*

1) При построении параметров кубического сплайна воспользоваться алгоритмом прогонки. Выполнить оценку погрешности аппроксимации значений первой производной в узлах интерполяции.

1. **Исходные данные**

|  |  |
| --- | --- |
| Вариант №3: | Вариант №6 |
| image003 |  |

1. **Решение**

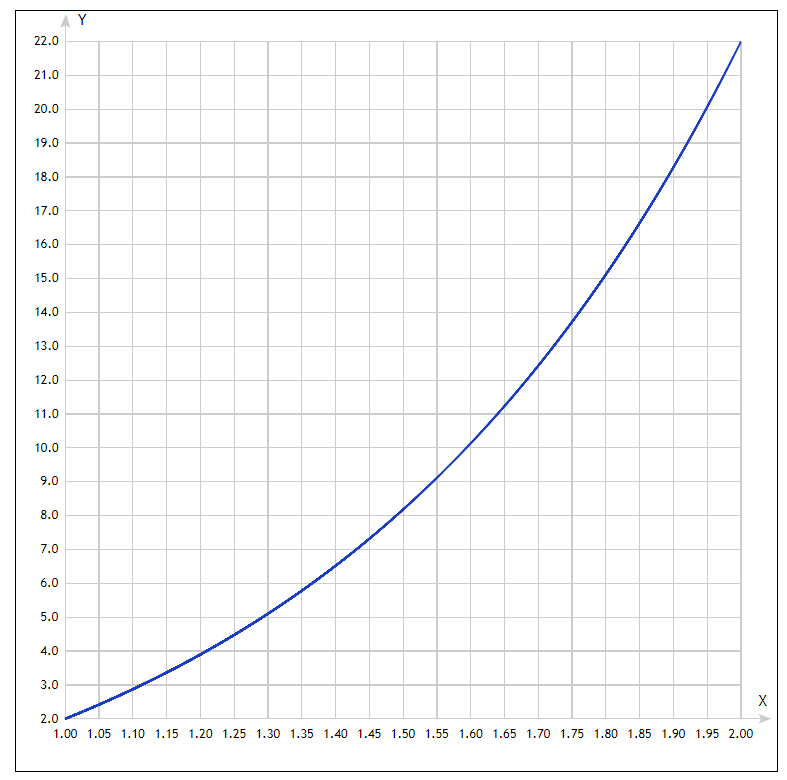
В данном разделе будут приведены результаты, а также подробные шаги выполнения программы, на примере двух уравнений: №3 и №6.

На вход программе ничего подавать не нужно, указанные варианты статически внесены в код программы. Программа автоматически генерирует файл «output.txt» и выводит в него все необходимые данные.

# **Набор №3.**

# image003

1. Воспользуемся возможностями сайта yotx.ru и построим график уравнения на [1,2]



Файл «output.txt»:

**ИСХОДНОЕ УРАВНЕНИЕ:**

f(x) = 5^x - 3

Построим таблицу точек и их значений:

xi f(xi)

1 2

1.2 3.89865

1.4 6.51827

1.6 10.1326

1.8 15.1195

2 22

**ИНТЕРПОЛЯЦИЯ ПО ФОРМУЛАМ НЬЮТОНА**

**ТАБЛИЦА РАЗДЕЛЁННЫХ РАЗНОСТЕЙ+**

xi f(xi) f(xi,xi+1) f(xi,..,xi+2) f(xi,..,xi+3) f(xi,..,xi+4) f(xi,..,xi+5)

1 2 9.49324 9.01216 5.70364 2.70730 1.02804

1.2 3.89865 13.0981 12.4343 7.86949 3.73535

1.4 6.51827 18.0718 17.1560 10.8578

1.6 10.1326 24.9343 23.6707

1.8 15.1195 34.4025

2 22

**ИНТЕРПОЛЯЦИОННАЯ ФОРМУЛА НЬЮТОНА**

Pn(x) = 2 + 9.493242(x-1) + 9.012163(x-1)(x-1.2) + 5.703643(x-1)(x-1.2)(x-1.4) + 2.707303(x-1)(x-1.2)(x-1.4)(x-1.6) + 1.028043(x-1)(x-1.2)(x-1.4)(x-1.6)(x-1.8)

**ОЦЕНКА ПОГРЕШНОСТЕЙ**

M6 = 434.496+

xi f(xi) Pb(x) Delta + Оценка+

1.1 2.87309 2.87333 0.000237223 0.000570275

1.3 5.10328 5.10320 8.25936e-05 0.000190092

1.5 8.18034 8.18040 6.17135e-05 0.000135780

1.7 12.4258 12.4258 9.05224e-05 0.000190092

1.9 18.2835 18.2838 0.000285001 0.000570275

**ИНТЕРПОЛЯЦИЯ КУБИЧЕСКИМИ СПЛАЙНАМИ ДЕФККТА 1**

Параметры кубических сплайнов: **m0** = 8.04719; **m5** = 40.2359+

Вектор правой части: (59.7269, 93.5099, 129.018, 137.774)

Решаем систему методом прогонки:

m1 = 11.1022; m2 = 15.3182; m3 = 21.1351; m4 = 29.1598 +

M5 = 269.967+

xi df/dx(x[i]) m[i] Delta + Оценка+

1 8.04719 8.04719 0 0.00719913

1.2 11.1029 11.1022 0.000769751 0.00719913

1.4 15.3191 15.3182 0.000913783 0.00719913

1.6 21.1362 21.1351 0.00108408 0.00719913

1.8 29.1622 29.1598 0.00235077 0.00719913

2 40.2359 40.2359 2.13163e-14 0.00719913

M4 = 167.74+

xi f(xi) S31(f;x) Abs(f(x)-S31(f;x)) + Оценка+

1.1 2.87309 2.87295 0.000145232 0.00105887

1.3 5.10328 5.10306 0.000223332 0.00105887

1.5 8.18034 8.18003 0.000308848 0.00105887

1.7 12.4258 12.4254 0.000400334 0.00105887

1.9 18.2835 18.2828 0.000654814 0.00105887

**СРЕДНЕКВАДРАТИЧНОЕ ПРИБЛИЖЕНИЕ**

**Дискретный вариант**

Выберем следующую систему функций:

{g0 = 1; g1 = x; g2 = x^2, т.о. g = C0 + C1x+ C2x^2}

Матрица: +

6 9 14.2

9 14.2 23.4

14.2 23.4 39.9664

Вектор правых частей: +

59.669 103.231 183.317

Решаем систему методом Гаусса:

Получили: (13.2695, -26.4313, 15.3474)

P2(x) = (13.2695) + (-26.4313)\*x + (15.3474)\*x^2 +

Оценка погрешности: 0.5235943**+**

**Непрерывный вариант+**

Матрица:

1 1.5 2.33333

1.5 2.33333 3.75

2.33333 3.75 6.2

Вектор правых частей: +

9.4267 15.7389 26.8765

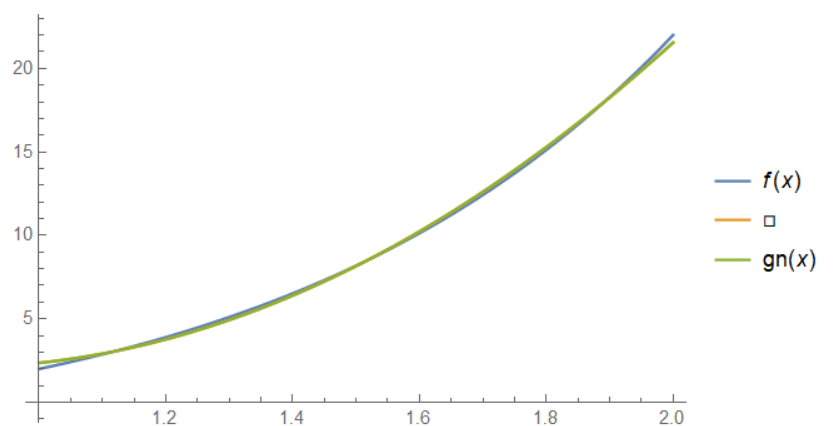
Решаем систему методом Гаусса:

Получили: (13.4981, -26.2998, 15.1621)

P2(x) = (13.4981) + (-26.2998)\*x + (15.1621)\*x^2+

Оценка погрешности: 0.15294215**+**

**ГРАФИКИ**



**МЕТОД ОБРАТНОГО ИНТЕРПОЛИРОВАНИЯ**

**Исходная таблица:**

xi f(xi)

1 2

1.2 3.89865

1.4 6.51827

1.6 10.1326

1.8 15.1195

2 22

**Преобразуем её к виду:**

yi xi

2 1

3.89865 1.2

6.51827 1.4

10.1326 1.6

15.1195 1.8

22 2

**ТАБЛИЦА РАЗДЕЛЁННЫХ РАЗНОСТЕЙ+**

2 1 0.105338 -0.00641644 0.000374521 -1.76782e-05 6.4e-07

3.89865 1.2 0.0763469 -0.00337059 0.000142592 -4.87824e-06

6.51827 1.4 0.0553347 -0.00177059 5.42891e-05

10.1326 1.6 0.0401055 -0.0009301

15.1195 1.8 0.0290676

22 2

Интерполяционная формула Ньютона

0.86617

1

1.11286

P5(y) 1.20911

1.29231

1.36536

c = 6

Корень: 1.36536 +

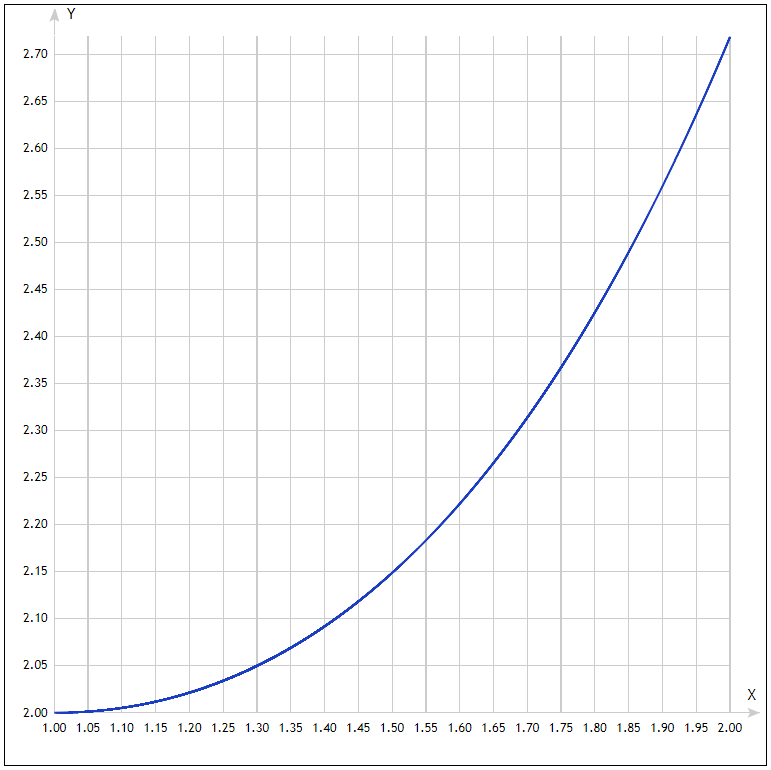
P5(1.36536) = 6.00221

Погрешность: 0.00220779+

# **Набор №6.**

# 

1. Воспользуемся возможностями сайта yotx.ru и построим график уравнения на [1,2]



Файл «output.txt»:

**ИСХОДНОЕ УРАВНЕНИЕ:**

f(x) = e^(x-1)+2-x

Построим таблицу точек и их значений:

xi f(xi)

1 2

1.2 2.0214

1.4 2.09182

1.6 2.22212

1.8 2.42554

2 2.71828

**ИНТЕРПОЛЯЦИЯ ПО ФОРМУЛАМ НЬЮТОНА**

**ТАБЛИЦА РАЗДЕЛЁННЫХ РАЗНОСТЕЙ +**

xi f(xi) f(xi,xi+1) f(xi,..,xi+2) f(xi,..,xi+3) f(xi,..,xi+4) f(xi,..,xi+5)

1 2 0.107014 0.612740 0.226104 0.0625750 0.0138543

1.2 2.02140 0.352110 0.748402 0.276164 0.0764293

1.4 2.09182 0.651471 0.914100 0.337307

1.6 2.22212 1.017110 1.116480

1.8 2.42554 1.463700

2 2.71828

**ИНТЕРПОЛЯЦИОННАЯ ФОРМУЛА НЬЮТОНА**

2 + 0.107014(x-1) + 0.612740(x-1)(x-1.2) + 0.226104(x-1)(x-1.2)(x-1.4) + 0.062575(x-1)(x-1.2)(x-1.4)(x-1.6) + 0.013854(x-1)(x-1.2)(x-1.4)(x-1.6)(x-1.8)

**ОЦЕНКА ПОГРЕШНОСТЕЙ**

M6 = 2.71828+

xi f(xi) Pb(x) Delta + Оценка+

1.1 2.00517 2.00517 2.0592e-06 3.56774e-06

1.3 2.04986 2.04986 7.05634e-07 1.18925e-06

1.5 2.14872 2.14872 5.18455e-07 8.49463e-07

1.7 2.31375 2.31375 7.47076e-07 1.18925e-06

1.9 2.55960 2.55961 2.30826e-06 3.56774e-06

**ИНТЕРПОЛЯЦИЯ КУБИЧЕСКИМИ СПЛАЙНАМИ ДЕФККТА 1**

Параметры кубических сплайнов: **m0** = 0; **m5** = 1.71828+

Вектор правой части: (1.37737, 3.01074, 5.00574, 5.72416)

Решаем систему методом прогонки:

m1 = 0.22139; m2 = 0.491812; m3 = 0.822104; m4 = 1.22551

M5 = 2.71828+

xi df/dx(x[i]) m[i] + Delta + Оценка+

1 0 0 0 7.24875e-05

1.2 0.221403 0.22139 1.30607e-05 7.24875e-05

1.4 0.491825 0.491812 1.3023e-05 7.24875e-05

1.6 0.822119 0.822104 1.45631e-05 7.24875e-05

1.8 1.22554 1.225510 2.60897e-05 7.24875e-05

2 1.71828 1.718280 4.44089e-16 7.24875e-05

M4 = 2.71828+

xi f(xi) S31(f;x) Abs(f(x)-S31(f;x)) + Оценка+

1.1 2.00517 2.00517 4.28143e-06 1.49506e-05

1.3 2.04986 2.04985 5.62910e-06 1.49506e-05

1.5 2.14872 2.14871 6.83575e-06 1.49506e-05

1.7 2.31375 2.31374 8.10807e-06 1.49506e-05

1.9 2.5596 2.55959 1.09074e-05 1.49506e-05

**СРЕДНЕКВАДРАТИЧНОЕ ПРИБЛИЖЕНИЕ**

**Дискретный вариант**

Выберем следующую систему функций:

{g0 = 1; g1 = x; g2 = x^2, т.о. g = C0 + C1x+ C2x^2}

Матрица: +

6 9 14.2

9 14.2 23.4

14.2 23.4 39.9664

Вектор правых частей: +

13.4792 20.7122 33.4313

Решаем систему методом Гаусса: +

Получили: (2.98797, -1.82462, 0.843166)

P2(x) = (2.98797) + (-1.82462)\*x + (0.843166)\*x^2

Оценка погрешности: 0.01801116**+**

**Непрерывный вариант**

Матрица: +

1 1.5 2.33333

1.5 2.33333 3.75

2.33333 3.75 6.2

Вектор правых частей: +

2.21828 3.38495 5.35323

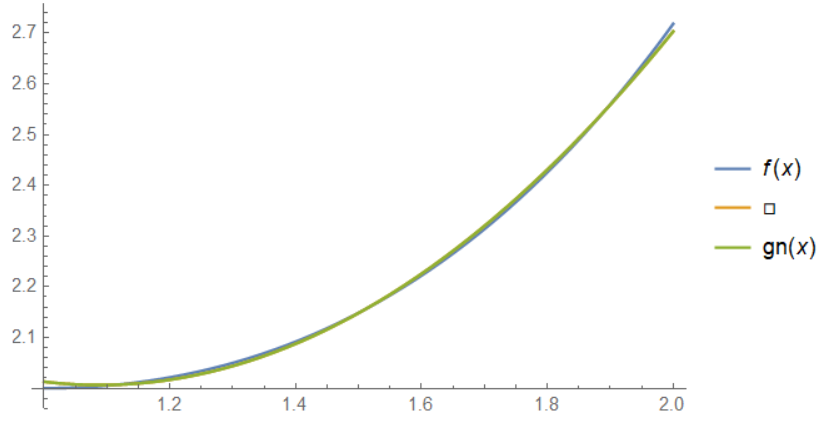
Решаем систему методом Гаусса:+

Получили: (3.00105, -1.82724, 0.839184)

P2(x) = (3.00105) + (-1.82724)\*x + (0.839184)\*x^2

Оценка погрешности: 0.005275926**+**

**ГРАФИКИ**



**МЕТОД ОБРАТНОГО ИНТЕРПОЛИРОВАНИЯ**

**Исходная таблица:**

xi f(xi)

1 2

1.2 2.0214

1.4 2.09182

1.6 2.22212

1.8 2.42554

2 2.71828

**Преобразуем её к виду:**

yi xi

2 1

2.0214 1.2

2.09182 1.4

2.22212 1.6

2.42554 1.8

2 2

**ТАБЛИЦА РАЗДЕЛЁННЫХ РАЗНОСТЕЙ+**

2 1 9.34459 -70.83680 289.642 -652.452 887.729

2.0214 1.2 2.84002 -6.501900 11.9968 -14.8123

2 .09182 1.4 1.53499 -1.653540 1.6744

2.22212 1.6 0.983177 -0.604598

2.42554 1.8 0.683198

2.71828 2

Интерполяционная формула Ньютона

-307.67

1

99.933

P5(y) 11931.2

124804

609214

c = 2.14872 **+**

P5(2.14872) = 1.43151**+**

Погрешность: 0.0406499 **+**

1. **Краткие выводы**

Получены формулы для приближения исходной функции по каждому из предложенных методов (формулы Ньютона, метод кубических сплайнов дефекта 1, среднеквадратическое приближение, обратное интерполирование).

По формулам Ньютона и методом среднеквадратичного приближения получили единую приближённую формулу для всего отрезка интерполирования, что иногда бывает более удобно, чем, как в случае с кубическими сплайнами, получать разные формулы для каждого из отрезков разбиения.

Кубические сплайны дают разные приближённые формулы для каждого из отрезков разбиения.

С точки зрения программирования данные методы не очень удобно программировать (приближенное вычисление интегралов, производных), но достаточно просто, конечно, если использовать математические языки, например, WolframMathematica, то будет проще и наглядней, но выбор пал на язык C++, реализация приведена в следующем разделе.

1. **Текст программы**

Программа состоит из одного файла «Main.cpp» - исходный код программы

#define \_USE\_MATH\_DEFINES

#include <fstream>

#include <iomanip>

#include <vector>

#include <functional>

#include <string>

#include <iostream>

using namespace std;

const int variant = 6;

//typedef vector<pair<double, double>> m5\_2;

typedef vector<double> myVector;

typedef vector<myVector> myMatrix;

const double EPS = 0.0001;

// Значение заданной функции в точке

double f(double x)

{

switch (variant)

{

case 1: return pow(2, x) + 5 \* x - 3;

case 3: return pow(5, x) - 3;

case 6: return pow(M\_E, x - 1) + 2 - x;

case 25: return pow(3, x) + 2 - x;

default: cout << "No function!"; return 0;

}

}

// Плохо, но работает (производная N-го порядка)

long double diffN(int n, long double x)

{

if (n == 1) return (f(x+EPS)-f(x))/(EPS);

else return ((diffN(n - 1, x + EPS) - diffN(n - 1, x)) / EPS);

}

// Вычисление производной n-го порядка

double Diff(int n, double x)

{

switch (variant)

{

case 1:

switch (n)

{

case 1: return pow(2, x)\*log(2) + 5;

case 2:case 3:сase 4:case 5:case 6:

return pow(2, x)\*pow(log(2), n);

}

case 3:

switch (n)

{

case 1: case 2:case 3:case 4:case 5:case 6:

return pow(5, x)\*pow(log(5), n);

}

case 6:

switch (n)

{

case 1: return pow(M\_E, x - 1)-1;

case 2:case 3:case 4:case 5:

case 6: return pow(M\_E, x-1);

}

case 25:

switch (n)

{

case 1: return pow(3, x)\*log(3) - 1;

case 2:case 3:ase 4: case 5:

case 6: return pow(3, x)\*pow(log(3), n);

}

default: cout << "No function!"; return 0;

}

}

// Создаем таблицу xi, yi

myMatrix CreateTable(double a, int n, double h, ofstream & fout)

{

//function<void(double a, int i, double h)> f = a + i \* h;

fout << setw(10) << "xi" << setw(10) << "f(xi)" << endl;

myVector vxi, vfxi;

for (int i = 0; i <= n; i++)

{

double xi = a + i\*h;

double fxi = f(xi);

vxi.push\_back(xi);

vfxi.push\_back(fxi);

fout << setw(10) << xi << setw(10) << fxi << endl;

}

return myMatrix { vxi, vfxi };

}

// Максимальная производная

double MaxDiff(int n, myMatrix table, int p, ofstream & fout)

{

double maxDiff(DBL\_MIN);

for (int i = 0; i <= n; i++)

{

auto curSixDiff = Diff(p, table[0][i]);

if (maxDiff < curSixDiff)

maxDiff = curSixDiff;

}

fout << "M" << p << " = " << maxDiff << endl << endl;

return maxDiff;

}

// Вычисление интерполяционного полинома Ньютона

double CalcPn(myMatrix table, myMatrix splDiffTable, int n, double x)

{

double Pn = table[1][0];

for (int i = 0; i <= n; i++)

{

if (i != n) {

double mult = splDiffTable[i + 1][0];

for (int k = 0; k <= i; k++)

mult \*= x - table[0][k];

Pn += mult;

}

}

return Pn;

}

// W(x) = (x-x0)(x-x1)...(x-xn) в нашем случае (x-1)(x-1.2)(x-1.4)(x-1.6)(x-1.8)(x-2)

double CalcW(myMatrix table, double x, int n)

{

double mult = 1;

for (int i = 0; i <= n; i++)

{

mult \*= x - table[0][i];

}

return abs(mult);

}

void NewtonMethod(myMatrix table, int n, ofstream & fout)

{

fout << "\n\nИНТЕРПОЛЯЦИЯ ПО ФОРМУЛАМ НЬЮТОНА\n\n";

const int fact6 = 720;

myMatrix splDiffTable;

splDiffTable.push\_back(table[1]);

for (int i = 0; i < n; i++)

{

vector <double> fx;

for (int j = 0; j < n - i; j++)

{

fx.push\_back((splDiffTable[i].at(j + 1) - splDiffTable[i].at(j)) / (table[0].at(j + i + 1) - table[0].at(j)));

}

splDiffTable.push\_back(fx);

}

fout << "ТАБЛИЦА РАЗДЕЛЁННЫХ РАЗНОСТЕЙ\n\n";

fout << setw(5) << "xi" << setw(18) << "f(xi)" << setw(18) << "f(xi,xi+1)" << setw(18) << "f(xi,..,xi+2)"

<< setw(18) << "f(xi,..,xi+3)" << setw(18) << "f(xi,..,xi+4)" << setw(18) << "f(xi,..,xi+5)\n";

string Pn = to\_string(table[1][0]) + " + ";

for (int i = 0; i <= n; i++)

{

fout << setw(5) << table[0][i];

if (i != n) {

Pn += to\_string(splDiffTable[i + 1][0]);

for (int k = 0; k <= i; k++)

Pn += "(x-" + to\_string(table[0][k]) + ")";

Pn += " + ";

}

for (int j = 0; j <= n - i; j++)

{

fout << setw(18) << splDiffTable[j][i];

}

fout << endl;

}

fout << "\nИНТЕРПОЛЯЦИОННАЯ ФОРМУЛА НЬЮТОНА\n\n";

fout << Pn;

// Поиск максимальной производной

auto maxDiff = MaxDiff(n, table, 6, fout);

fout << "\nОЦЕНКА ПОГРЕШНОСТЕЙ\n\n";

fout << setw(5) << "xi" << setw(18) << "f(xi)" << setw(18) << "Pb(x)" << setw(18) << "Delta" << setw(18) << "Оценка\n";

for (double x = 1.1; x < 2; x += 0.2)

{

auto fxi = f(x);

auto Pn = CalcPn(table, splDiffTable, n, x);

auto wi = CalcW(table, x, n);

// delta = f(xi) - Pn(xi) // ocenka = Mn+1 \* |Wn+1| / (n+1)!

fout << setw(5) << x << setw(18) << fxi << setw(18) << Pn << setw(18) << abs(fxi - Pn) << setw(18) << maxDiff \* wi / fact6 << endl;

}

}

// Метод прогонки

myVector SweepMethod(myMatrix m, myVector r, int const n)

{

double y;

myVector a(n+1), B(n+1), resV(n+1);

y = m[0][0];

a[0] = -m[0][1] / y;

B[0] = r[0] / y;

for (int i = 1; i < n; i++) {

y = m[i][i] + m[i][i - 1] \* a[i - 1];

a[i] = -m[i][i + 1] / y;

B[i] = (r[i] - m[i][i - 1] \* B[i - 1]) / y;

}

resV[n] = (r[n] - m[n][n - 1] \* B[n - 1]) / (m[n][n] + m[n][n - 1] \* a[n - 1]);

for (int i = n - 1; i >= 0; i--)

{

resV[i] = a[i] \* resV[i + 1] + B[i];

}

return resV;

}

double Phi0(double t)

{

return pow((1 - t), 2) \* (1 + 2 \* t);

}

double Phi1(double t)

{

return pow((1 - t), 2) \* t;

}

double CalcS31(myVector b, myMatrix table, double x, int i, double h)

{

double xi = table[0][i], xinext = table[0][i+1];

double t = (x - xi)/h;

return Phi0(t)\*f(xi) + Phi0(1 - t)\*f(xinext) + h \* (Phi1(t)\*b[i] - Phi1(1 - t)\*b[i + 1]);

}

void CubicSpline(myMatrix table, int n, double a, double b, double h, ofstream & fout)

{

fout << "\n\nИНТЕРПОЛЯЦИЯ КУБИЧЕСКИМИ СПЛАЙНАМИ ДЕФККТА 1\n\n";

double b0 = Diff(1, a);

double b5 = Diff(1, b);

myMatrix m{ {4,1,0,0}, {1,4,1,0}, {0,1,4,1}, {0,0,1,4}};

fout << "Параметры кубических сплайнов: \nb0 = " << b0 << "; b5 = " << b5 << endl;

myVector r;

for (int i = 1; i < n; i++)

{

r.push\_back(3\*(f(table[0][i+1]) - f(table[0][i-1]))/h);

}

r[0] -= b0;

r[3] -= b5;

fout << "Вектор правой части: ";

for (int i = 0; i < n-1; i++)

fout << r[i] << " ";

fout << "Решаем систему методом прогонки: \n";

auto B = SweepMethod(m, r, n-2);

for (int i = 0; i < n-1; i++)

fout << "b" << i+1 << " = " << B[i] << " ";

fout << endl << endl;

myVector b1;

b1.push\_back(b0);

for (int i = 0; i < 4; i++)

b1.push\_back(B.at(i));

b1.push\_back(b5);

auto maxDiff5 = MaxDiff(n, table, 5, fout);

fout << setw(5) << "xi" << setw(18) << "df/dx(x[i])" << setw(18) << "m[i]" << setw(18) << "Delta" << setw(18) << "Оценка\n";

int i = 0;

for (double x = 1; x <= 2; x += 0.2)

{

auto dfxi = Diff(1, x);

fout << setw(5) << x << setw(18) << dfxi << setw(18) << b1[i] << setw(18) << abs(dfxi - b1[i]) << setw(18) << pow(h, 4) \* maxDiff5 / 60 << endl;

i++;

}

auto maxDiff4 = MaxDiff(n, table, 4, fout);

fout << endl << setw(5) << "xi" << setw(18) << "f(xi)" << setw(18) << "S31(f;x)" << setw(18) << "Delta" << setw(18) << "Оценка\n";

i = 0;

for (double x = 1.1; x < 2; x += 0.2)

{

auto fxi = f(x);

auto S31 = CalcS31(b1, table, x, i, h);

i++;

fout << setw(5) << x << setw(18) << fxi << setw(18) << S31 << setw(18) << abs(fxi - S31) << setw(18) << pow(h, 4) \* (maxDiff4 / 384 + maxDiff5 \* h / 240) << endl;

}

}

double g(double x, int n)

{

switch (n)

{

case 0: return 1;

case 1: return x;

case 2: return x \* x;

}

}

myVector gauss(myMatrix a, myVector y, int n)

{

double max;

int k, index;

const double eps = 0.00001; // точность

myVector x(n);

k = 0;

while (k < n)

{

// Поиск строки с максимальным a[i][k]

max = abs(a[k][k]);

index = k;

for (int i = k + 1; i < n; i++)

{

if (abs(a[i][k]) > max)

{

max = abs(a[i][k]);

index = i;

}

}

// Перестановка строк

if (max < eps)

{

// нет ненулевых диагональных элементов

cout << "Решение получить невозможно из-за нулевого столбца ";

cout << index << " матрицы A" << endl;

return { 0 };

}

for (int j = 0; j < n; j++)

{

double temp = a[k][j];

a[k][j] = a[index][j];

a[index][j] = temp;

}

double temp = y[k];

y[k] = y[index];

y[index] = temp;

// Нормализация уравнений

for (int i = k; i < n; i++)

{

double temp = a[i][k];

if (abs(temp) < eps) continue; // для нулевого коэффициента пропустить

for (int j = 0; j < n; j++)

a[i][j] = a[i][j] / temp;

y[i] = y[i] / temp;

if (i == k) continue; // уравнение не вычитать само из себя

for (int j = 0; j < n; j++)

a[i][j] = a[i][j] - a[k][j];

y[i] = y[i] - y[k];

}

k++;

}

// обратная подстановка

for (k = n - 1; k >= 0; k--)

{

x[k] = y[k];

for (int i = 0; i < k; i++)

y[i] = y[i] - a[i][k] \* x[k];

}

return { x };

}

double IntegrF(double a, double b)

{

// вычисляем интеграл по формуле трапеций

double eps = 0.000001;

double Integral = eps \* (f(a)\*f(a) + f(b)\*f(b)) / 2.0;

for (int i = 1; i <= (b - a) / eps - 1; i++)

Integral = Integral + eps \* f(a + eps \* i)\*f(a + eps \* i);

return Integral;

}

double IntegrG(double a, double b, int n)

{

// вычисляем интеграл по формуле трапеций

double eps = 0.000001;

double Integral = eps \* (f(a)\*g(a, n) + f(b)\*g(b, n)) / 2.0;

for (int i = 1; i <= (b - a) / eps - 1; i++)

Integral = Integral + eps \* f(a + eps \* i)\*g(a + eps \* i, n);

return Integral;

}

void RMS\_Approximation(myMatrix table, int n, double a, double b, ofstream & fout)

{

fout << "\n\nСРЕДНЕКВАДРАТИЧЕСКОЕ ПРИБЛИЖЕНИЕ\n\n";

fout << "\nДискретный вариант\n";

fout << "Выберем следующую систему функций: {g0 = 1; g1 = x; g2 = x^2, т.о. g = C0 + C1x+ C2x^2}";

myMatrix matrixScalarD{ {6., 9., 14.2}, {9., 14.2, 23.4}, {14.2, 23.4, 39.9664} };

myVector r;

double G = 0, F = 0;

for (int i = 0; i < 3; i++)

{

double sum = 0;

for (double x = 1; x <= 2; x+=0.2)

{

sum += g(x, i)\*f(x);

}

r.push\_back(sum);

}

fout << "Матрица: \n";

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

fout << setw(13) << matrixScalarD[i][j];

fout << endl;

}

fout << "Вектор правых частей: \n";

for (int j = 0; j < 3; j++)

fout << setw(13) << r[j];

fout << endl;

fout << "Решаем систему: \n";

auto B = gauss(matrixScalarD, r, 3);

fout << "Получили: \n" << B[0] << " " << B[1] << " " << B[2] << endl << endl;

fout << "P2(x) = (" << B[0] << ") + (" << B[1] << ")\*x + (" << B[2] << ")\*x^2 ";

for (double x = 1; x <= 2; x+=0.2)

{

F += (f(x) \* f(x));

G += ((B[0] + B[1] \* x + B[2] \* x\*x)\*(B[0] + B[1] \* x + B[2] \* x\*x));

}

fout << "\nОценка погрешности: " << sqrt(F - G) << endl << endl;

fout << "\nНепрерывный вариант\n";

myMatrix matrixScalarN{ {1., 1.5, 2.33333}, {1.5, 2.33333, 3.75}, {2.33333, 3.75, 6.2} };

myVector r1{ IntegrG(a,b,0), IntegrG(a,b,1), IntegrG(a,b,2) };

fout << "Матрица: \n";

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

fout << setw(13) << matrixScalarN[i][j];

fout << endl;

}

fout << "Вектор правых частей: \n";

for (int j = 0; j < 3; j++)

fout << setw(13) << r1[j];

fout << endl;

fout << "Решаем систему: \n";

auto B1 = gauss(matrixScalarN, r1, 3);

fout << "Получили: \n" << B1[0] << " " << B1[1] << " " << B1[2] << endl << endl;

fout << "P2(x) = (" << B1[0] << ") + (" << B1[1] << ")\*x + (" << B1[2] << ")\*x^2 ";

F = IntegrF(a,b) \* IntegrF(a,b);

//fout << "\nОценка погрешности: " << F - Integral << endl << endl;

}

void ReverseInterpolation(myMatrix table, int n, double a, double b, double h, ofstream & fout)

{

myMatrix m(8, {0, 0, 0, 0, 0, 0, 0, 0});

for (int i = 0; i <= n; i++)

{

m[i + 2][1] = f(a+h\*i);

m[i + 2][2] = a + i\*h;

}

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n - i; j++)

{

m[j + 2][i + 3] = (m[j + 3][i + 2] - m[j + 2][i + 2]) / (m[i + j + 3][1] - m[j + 2][1]);

}

}

myVector y(table[1]);

fout << "ТАБЛИЦА РАЗДЕЛЁННЫХ РАЗНОСТЕЙ\n\n";

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

fout << setw(18) << m[i][j];

}

fout << endl;

}

double c = f(1.5);

fout << c << endl;

double P5 = m[0][2] + m[0][3] \* (c - y[0]) + m[0][4] \* (c - y[0])\*(c - y[1]) + m[0][5] \* (c - y[0])\*(c - y[1])\*(c - y[2]) +

m[0][6] \* (c - y[0])\*(c - y[1])\*(c - y[2])\*(c - y[3]) + m[0][7] \* (c - y[0])\*(c - y[1])\*(c - y[2])\*(c - y[3])\*(c - y[4]);

fout << endl << P5 << " " << f(P5) << " " << f(P5) - c << endl;}

int main()

{

const int n = 5;

const double a = 1, b = 2, h = 0.2; // h = (b-a)/n

ofstream fout("output.txt");

myMatrix table = CreateTable(a, n, h, fout);

NewtonMethod(table, n, fout);

CubicSpline(table, n, a, b, h, fout);

RMS\_Approximation(table, n, a, b, fout);

ReverseInterpolation(table, n, a, b, h, fout);

}