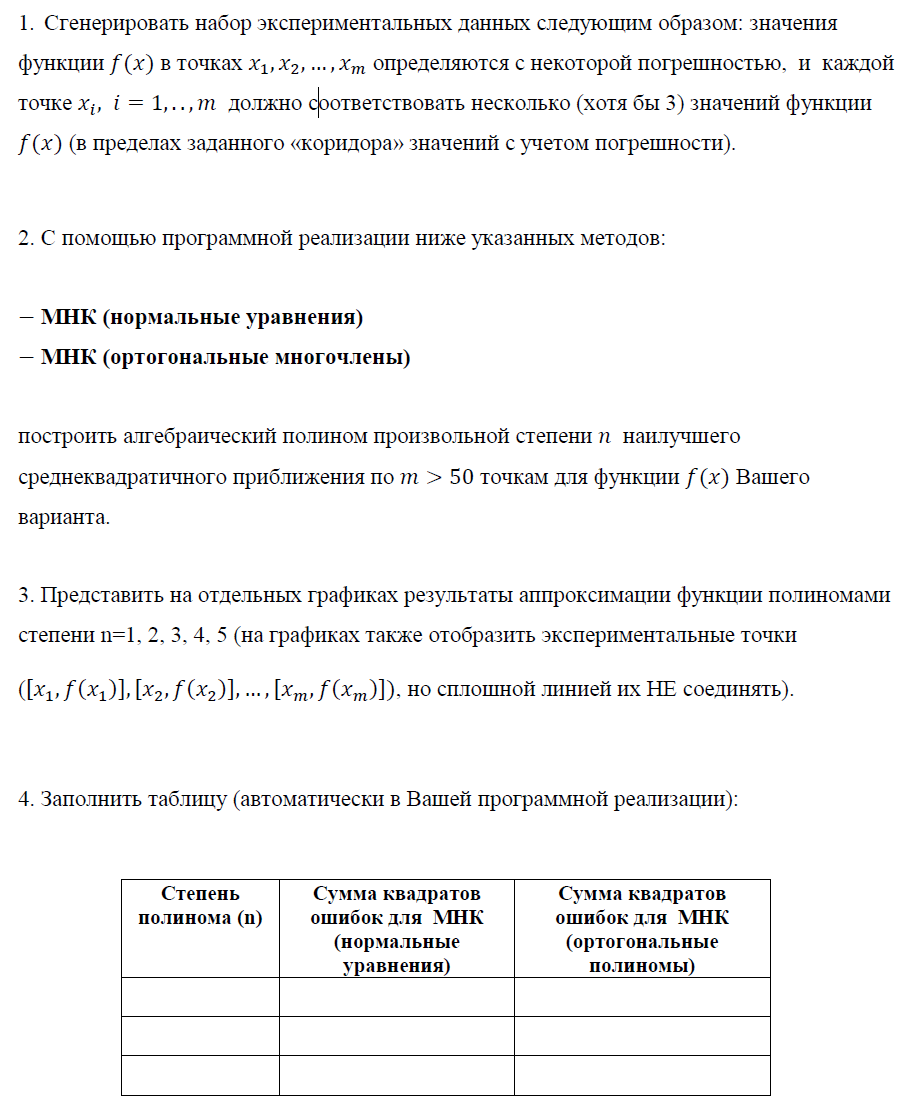
**Задание №3. Аппроксимация функций**

***Цель задания***: *практическое освоение метода наименьших квадратов (МНК) для аппроксимации функции*



Код программы

#include <iostream>

#include <stdio.h>

#include <conio.h>

#include <locale>

#include<fstream>

#include <string>

#include <stdlib.h>

#include <cmath>

#include <ctime>

#define N 5//максимальная степень полинома

#define M 303//число точек (101 точек по 3 значения в каждой)

using namespace std;

double x[M], y[M], F[M];

double aa = -1, bb =1, SSE1 = 0, SSE2 = 0;

//для обычных полиномов

double Am[N+1][N+1], Bm[N+1], Coef[N+1][M], P[N+1][M];

//для ортогональных полиномов

double q[N+1][M], alpha[N+1], beta[N+1], A[N+1], PN[N+1][M];

//интерполируемая функция

double f(double x)

{

return x\*x\*sin(x);

}

//определение равномерной сетки

void GetRavn()

{

int nn = M;

double h = (bb-aa)/(nn-3);

double kor = 0.1;//коридор для погрешности

//srand(time(NULL));//если добавить, то случайные числа будут всегда разные

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

{

x[i]=aa+h\*i;x[i+1]=x[i];x[i+2]=x[i];

double t = f(x[i]);

F[i] = t;//точное значение функции

//три значения в одной точке с погрешностями

double R = (double)(rand()%10000)/10000; // случайное число от 0 до 1

y[i]=t+kor\*(R-0.5);

R = (double)(rand()%10000)/10000;

y[i+1]=t+kor\*(R-0.5);

R = (double)(rand()%10000)/10000;

y[i+2]=t+kor\*(R-0.5);

}

}

//Решение системы классическим методом Гаусса

double\* Gauss(int NN, double (\*A)[N+1],double B[])

{

int nx=NN, i, j, k;

double d, s;

double \*X = new double[nx];//столбец с решением

for (k = 0; k < nx; k++) // прямой ход

{

for (j = k + 1; j < nx; j++)

{

if (A[j][k]!=0)

{

d = A[j][k] / A[k][k];

for (i = k; i < nx; i++)

{

A[j][i] = A[j][i] - d \* A[k][i];

}

B[j] = B[j] - d \* B[k];

}

}

}

for (k = nx-1; k >= 0; k--) // обратный ход

{

d = 0;

for (j = k + 1; j < nx; j++)

{

s = A[k][j] \* X[j];

d = d + s;

}

X[k] = (B[k] - d) / A[k][k];

}

return X;

}

//опредление коэфф MNK по нормальной системе

void SetCoefNormal(double x[], double y[])

{

//определяем суммы

int k = 2\*N+1;

double \*sum = new double[k];

double \*sumy = new double[N+1];

sum[0]=M;

for(int j = 1; j<k; j++)//суммы x в степени j

{

sum[j] = 0;

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

{

sum[j] = sum[j] + pow(x[i],j);

}

}

//определяем коэффициенты нормальной системы

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

{

Bm[j] = 0;

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

{

Bm[j] = Bm[j] + pow(x[i],j)\*y[i];

}

}

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

{

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

{

Am[i][j] = sum[i+j];

}

}

//для многочлена 0-го порядка на всякий случай

for(int i = 0; i<=k; i++) Coef[0][i] = sum[0]/M;

for(int i = 0; i<M; i++) P[0][i] = sum[0]/M;

//нахождение коэффициентов полиномов степени от 1 до 5

for(int k = 1; k<=N; k++)

{

double \*cf = new double[N+1];//столбец коэффициентов

cf = Gauss(k+1, Am, Bm);//решаем нормальную с-му

for(int i = 0; i<=k; i++) Coef[k][i] = cf[i];

delete [] cf;

}

delete [] sum;delete [] sumy;

//нахождение значений в точках сетки

for(int k = 1; k<=N; k++)

{

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

{

P[k][i] = Coef[k][0];

for(int j = 1; j<=k; j++)

{

P[k][i] = P[k][i] + Coef[k][j]\*pow(x[i],j);

}

}

}

}

//нахождение коэффициентов для ортогональных полиномов

//и значений полиномов в точках сетки

void SetABQ(double x[])

{

double t;

alpha[0] = 0; beta[0]=0;

double sx = 0;

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

sx = sx + x[i];

alpha[1] = sx/M;

//q[j][i]-оттогональный многочлен степени j в точке x[i]

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

{

q[0][i]=1;

q[1][i]=x[i]-alpha[1];

}

for(int j = 1; j<N; j++)

{

double saV = 0,saN = 0;//числитель и знаменатель в формуле для alpha (5.2.48)

double sbV = 0,sbN = 0;//числитель и знаменатель в формуле для beta (5.2.49)

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

{

t = q[j][i]\*q[j][i];

saV = saV + x[i]\*t;

saN = saN + t;

sbV = sbV + x[i]\*q[j][i]\*q[j-1][i];

sbN = sbN + q[j-1][i]\*q[j-1][i];

}

alpha[j+1] = saV/saN;

beta[j] = sbV/sbN;

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

{

q[j+1][i]= x[i]\*q[j][i]-alpha[j+1]\*q[j][i]-beta[j]\*q[j-1][i];

}

}

}

//опредление коэфф ортогональных полиномов по MNK и значений в точках

void SetCoef(double x[], double y[])

{

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

{

double saV = 0,saN = 0;//числитель и знаменатель в формуле для alpha (5.2.38)

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

{

saV = saV + q[j][i]\*y[i];

saN = saN + q[j][i]\*q[j][i];

}

A[j] = saV/saN;//коэфф. ортогонального полинома

}

//нахождение значений в точках сетки

for(int k = 1; k<=N; k++)

{

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

{

PN[k][i] = A[0];

for(int j = 1; j<=k; j++)

{

PN[k][i] = PN[k][i] + A[j]\*q[j][i];

}

}

}

}

//сумма квадратов ошибок

void SSE()

{

cout << "Степень полинома" << "\t"<<"SSE (нормальные уравнения)" << "\t"<<"SSE(ортогональные полиномы)"<<endl;

for(int k = 1; k<=N; k++)

{

SSE1 = 0;SSE2 = 0;

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

{

SSE1 = SSE1 + (y[i]-P[k][i])\*(y[i]-P[k][i]);

SSE2 = SSE2+ (y[i]-PN[k][i])\*(y[i]-PN[k][i]);

}

cout << k << "\t\t\t " << SSE1<< "\t\t\t" << SSE2 <<endl;

}

}

//печать таблицы

void PrintTable()

{

//вывод в файл

ofstream fout("output.txt");

fout << "x" << "\t"<<"f(x)" <<"\t"<<"P1(x)"<<"\t"<<"P2(x)"<<"\t"<<"P3(x)"<<"\t"<<"P4(x)"<<"\t"<<"P5(x)"<<"\t";

fout <<"Q1(x)"<<"\t"<<"Q2(x)"<<"\t"<<"Q3(x)"<<"\t"<<"Q4(x)"<<"\t"<<"Q5(x)" <<endl;

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

{

fout << x[i] << "\t" << y[i] << "\t";

for(int k = 1; k<=N; k++) fout << P[k][i] << "\t";

for(int k = 1; k<=N; k++) fout << PN[k][i] << "\t";

fout << endl;

}

fout.close();

}

int main()

{

setlocale(LC\_ALL, "russian");

GetRavn();

SetCoefNormal(x,y);

SetABQ(x);

SetCoef(x,y);

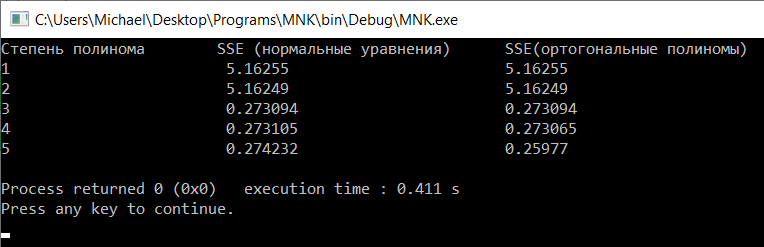
SSE();

PrintTable();

return 0;

}

Результаты выполнения



Выводы

Для данной функции хорошее приближения получается для полиномов 3-ей степени. Дальнейшее увеличение порядка полинома не приводит к значительному уменьшению суммы квадратов ошибки. Разложение по нормальным и ортогональным многочленам должно полностью совпадать, однако при вычислении разложения по нормальным многочленам есть вычислительная погрешность.

Так наблюдаем, что при степени многочлена больше 3 сумма квадратов ошибок не уменьшается. Это связано с тем, что суммы при нечётных степенях близки к нулю (исходная функция нечётная), из-за чего число обусловленности матрицы системы нормальных уравнений возрастает, отсюда погрешность при решении методом Гаусса.

Таким образом, использование ортогональных многочленов позволяет снизить вычислительную погрешность. Также плюсом является возможность строить полином степень за степенью.

Графики