«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет електроніки

Кафедра мікроелектроніки

Лабораторна робота №17

**Апроксимація функціональних залежностей методом найменших квадратів**

Варіант №21

Виконав: студент групи ДП-82

Мнацаканов Антон

Перевірив: Домбругов М.Р.

Київ-2020

**Мета роботи:**застосування методу найменших квадратів для побудови поліноміального середньоквадратичного наближення функції.

**Що зробити:**побудувати наближення функції f(x)лінійним поліномом за методом найменших квадратів, використовуючи набір із nпсевдовипадкових «експериментальних» точок (xі,yі)(i=1,2,...n), що розташовані на кривій f(x) та імітують серію вимірювань.Дослідити поведінку коефіцієнтів МНК-полінома та средньоквадратичнупохибкунаближення σ в залежності від числа точок n. Провести аналогічні дослідження, додавши до «експериментальних» точок випадкові похибки. Додатково – побудувати МНК-наближення поліномами більш високих степенів та дослідити залежність σвід кількості точокnта степеніМНК-полінома m.

**Хід роботи**

Код на С:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include<time.h>

double a=-1.0;

double b=3.0;

double func(double x){

return x\*x\*10.0\*exp(-x)-x\*3.0;

}

double frandom(){

return ((double)(rand()))/RAND\_MAX;

}

double\*\* gen\_points\_set(int n, double a, double b, double(\*func)(double)){

double\*\* res = malloc(n\*sizeof(double\*));

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

res[i] = malloc(2\*sizeof(double));

res[i][0] = a+(b-a)\*frandom();

res[i][1] = func(res[i][0]);

}

return res;

}

double\*\* gen\_points\_W\_set(int n, double a, double b, double(\*func)(double), double W){

double\*\* res = malloc(n\*sizeof(double\*));

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

res[i] = malloc(2\*sizeof(double));

res[i][0] = a+(b-a)\*frandom();

res[i][1] = func(res[i][0]) + (frandom()\*2-1)\*W;

}

return res;

}

double\* gauss(int n, double\*\* slar, double\* r){

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

int m =i;

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

if(fabs(slar[k][i])>fabs(slar[m][i]))

m=k;

}

if(m!=i){

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

double t=slar[m][j];

slar[m][j]=slar[i][j];

slar[i][j]=t;

}

double t=r[m];

r[m]=r[i];

r[i]=t;

}

if (slar[i][i]==0){

return NULL;

}

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

double p=slar[k][i]/slar[i][i];

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

slar[k][j]=slar[k][j]-p\*slar[i][j];

}

r[k]=r[k]-p\*r[i];

}

}

double\* result = malloc(n\*sizeof(double));

result[n-1]=r[n-1]/slar[n-1][n-1];

for (int i = n-2; i>=0; --i){

double s=0;

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

s=s+slar[i][j]\*result[j];

}

result[i]=(r[i]-s)/slar[i][i];

}

return result;

}

double x(int pow, int n, double\*\* points){

if(0 == pow){

return n;

}

double res = 0;

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

double s = 1;

for(int j=0;j<pow; ++j){

s\*=points[i][0];

}

res += s;

}

return res;

}

double xy(int pow, int n, double\*\* points){

double res = 0;

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

double s = points[i][1];

for(int j=0;j<pow; ++j){

s\*=points[i][0];

}

res += s;

}

return res;

}

double\* polinom\_aproximation(int n, double\*\* points, int power){

double\*\* slar =malloc((power+1)\*sizeof(double\*));

for(int i=0; i<power+1; ++i){

slar[i] = malloc((power+1)\*sizeof(double));

}

for(int i=0; i<power+1; ++i){

for(int j=0; j<power+1; ++j){

slar[i][j] = x(i+j, n, points);

}

}

double\* r = malloc((power+1)\*sizeof(double));

for(int i=0; i<power+1; ++i){

r[i] = xy(i, n, points);

}

return gauss(power+1, slar, r);

}

double aprox\_func(double x, int pow, double\* koef){

double res = koef[0];

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

double s = koef[i];

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

s\*=x;

res+=s;

}

return res;

}

void make(int points\_count, int power){

char s[256];

sprintf(s,"file\_%d\_%d.txt", points\_count, power);

FILE\* f = fopen(s, "w");

double \*\* points = gen\_points\_set(points\_count, a, b, func);

double\* aprox\_koef=polinom\_aproximation(points\_count, points, power);

double mean\_squared\_error = 0;

for(double s=a; s<=b; s+=0.001){

double fn = func(s);

fprintf(f, "%e %e", s, fn);

double ap = aprox\_func(s, power, aprox\_koef);

mean\_squared\_error += (ap-fn)\*(ap-fn);

fprintf(f, " %e", ap);

fprintf(f, "\n");

}

fclose(f);

sprintf(s,"file\_W\_%d\_%d.txt", points\_count, power);

f = fopen(s, "w");

double \*\* pointsW = gen\_points\_W\_set(points\_count, a, b, func, sqrt(mean\_squared\_error/points\_count)\*3);

double\* aprox\_koefW=polinom\_aproximation(points\_count, pointsW, power);

double mean\_squared\_errorW = 0;

for(double s=a; s<=b; s+=0.001){

double fn = func(s);

fprintf(f, "%e %e", s, fn);

double ap = aprox\_func(s, power, aprox\_koefW);

mean\_squared\_errorW += (ap-fn)\*(ap-fn);

fprintf(f, " %e", ap);

fprintf(f, "\n");

}

fclose(f);

printf("%d\t%d\t", points\_count, power);

for(int i=0; i<power+1; ++i){

printf("%e\t", aprox\_koef[i]);

}

for(int i=0; i<power+1; ++i){

printf("%e\t", aprox\_koefW[i]);

}

printf("%e\t%e\n", sqrt(mean\_squared\_error/points\_count), sqrt(mean\_squared\_errorW/points\_count));

}

int main(){

srand(time(0));

make(5, 1);

make(5\*3, 1);

make(5\*10, 1);

make(5\*30, 1);

make(5\*100, 1);

make(5, 2);

make(5\*3, 2);

make(5\*10, 2);

make(5\*30, 2);

make(5\*100, 2);

make(5, 3);

make(5\*3, 3);

make(5\*10, 3);

make(5\*30, 3);

make(5\*100, 3);

make(5, 4);

make(5\*3, 4);

make(5\*10, 4);

make(5\*30, 4);

make(5\*100, 4);

make(5, 5);

make(5\*3, 5);

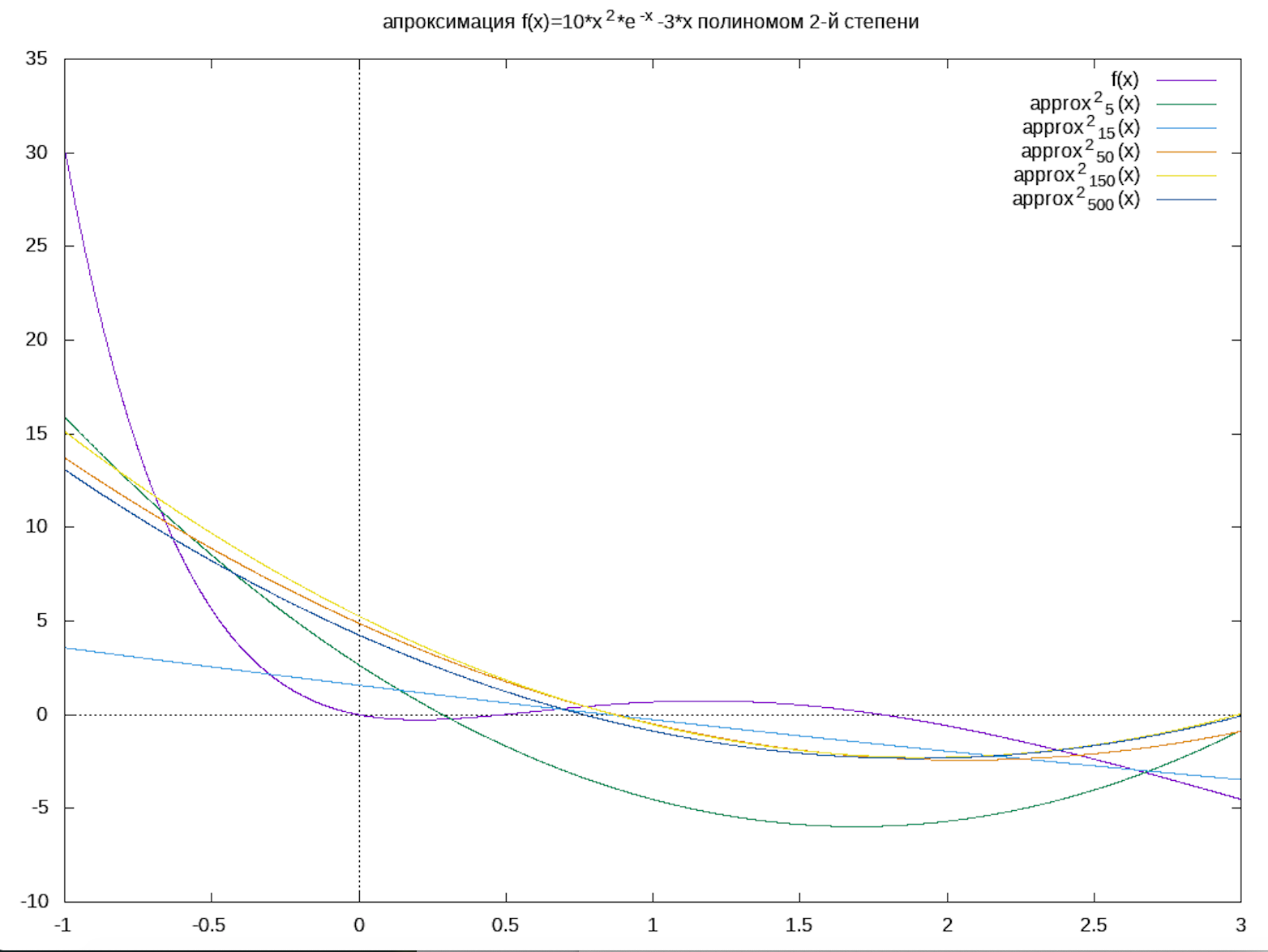
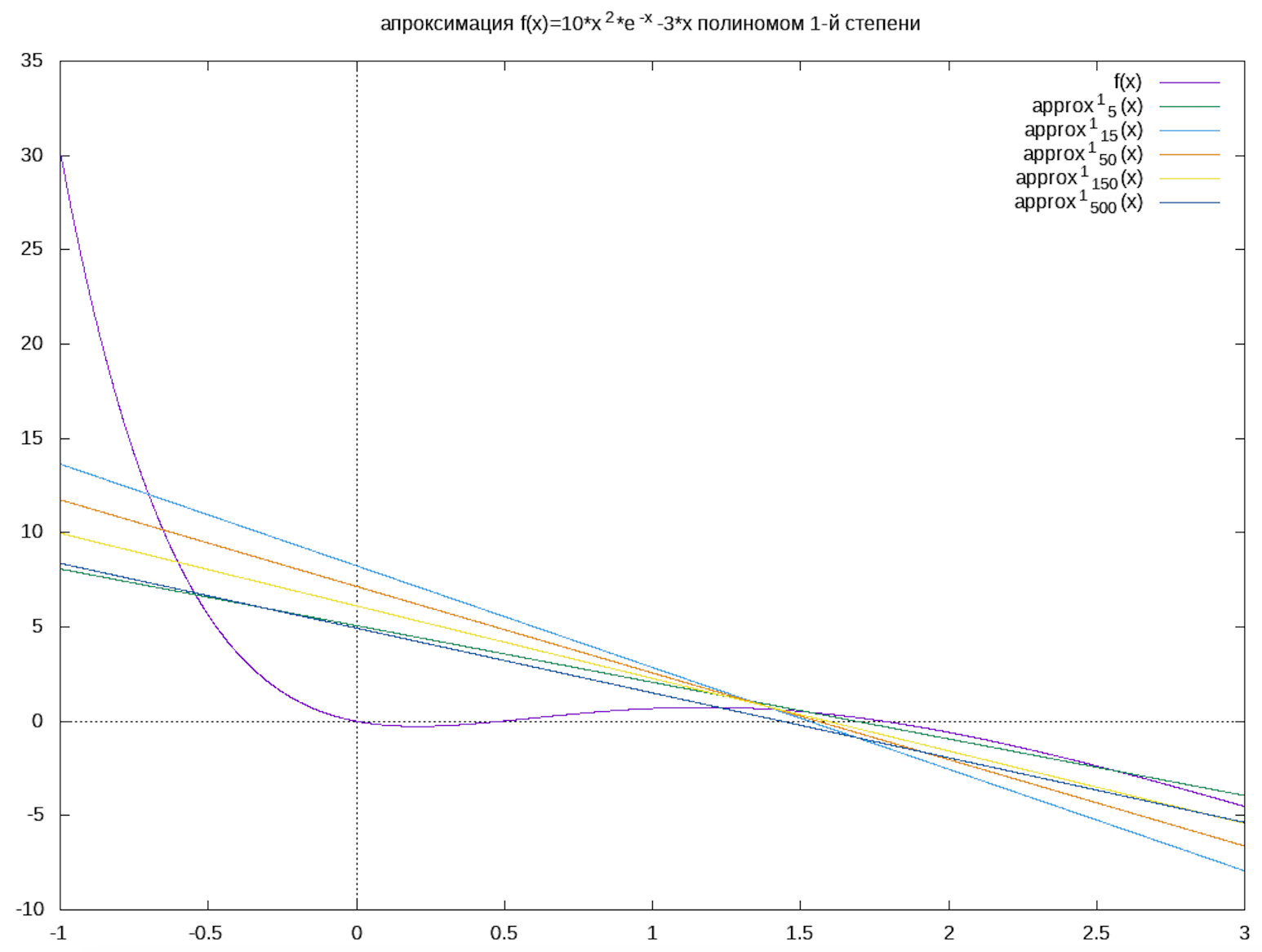
make(5\*10, 5);

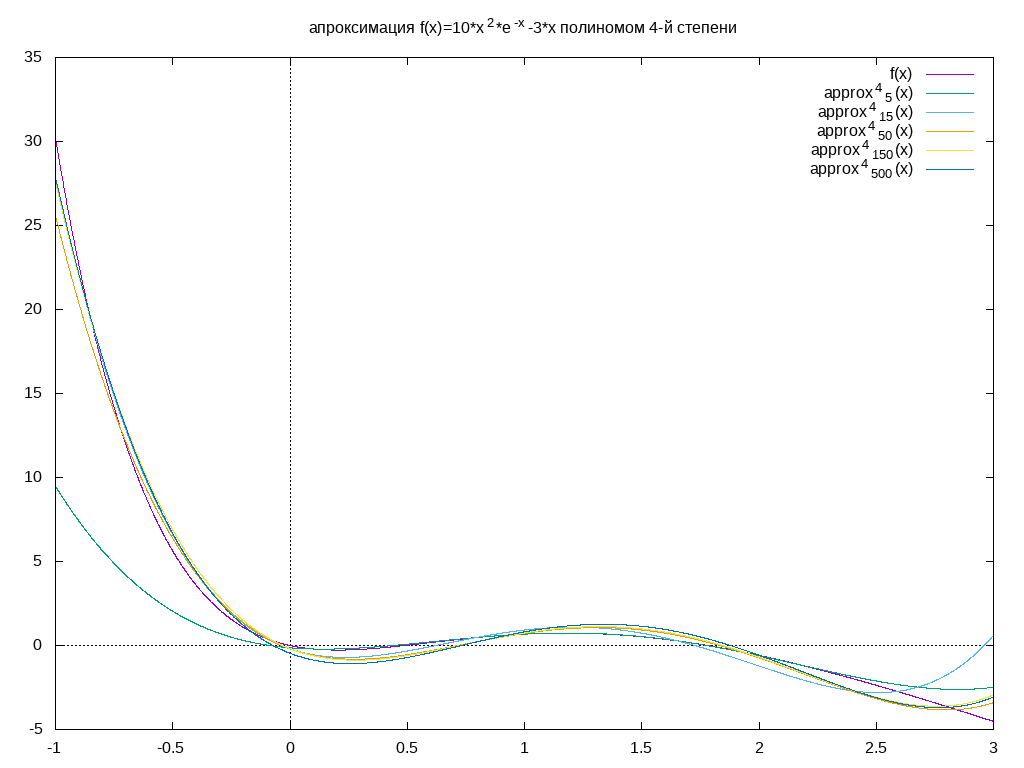
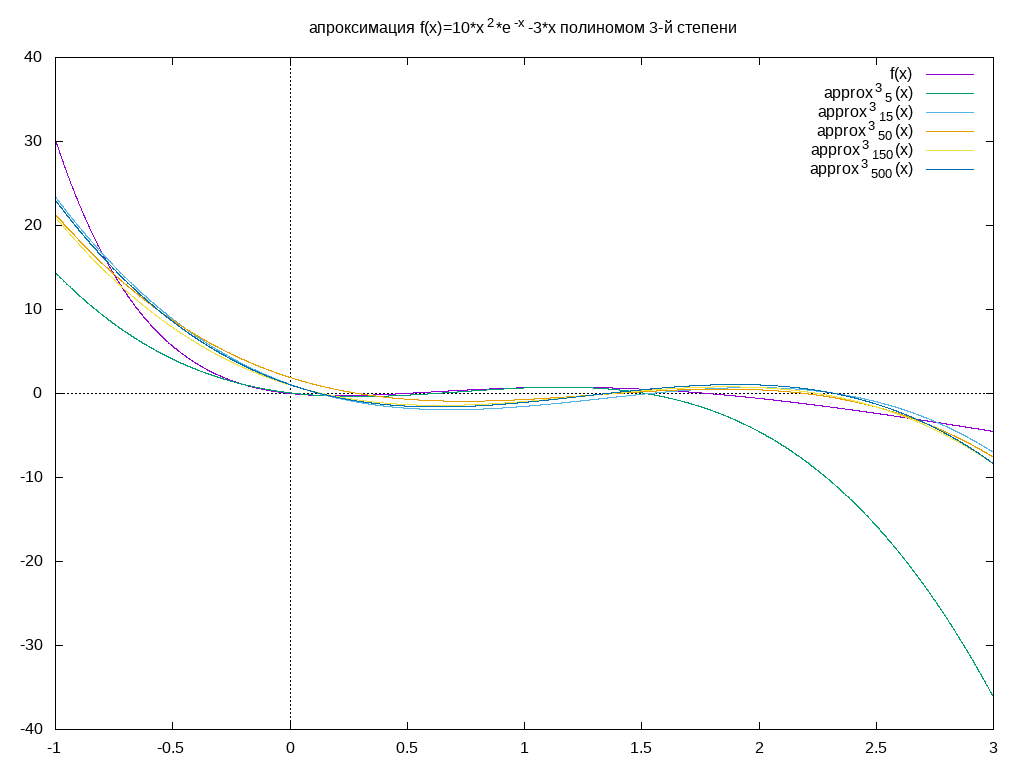
make(5\*30, 5);

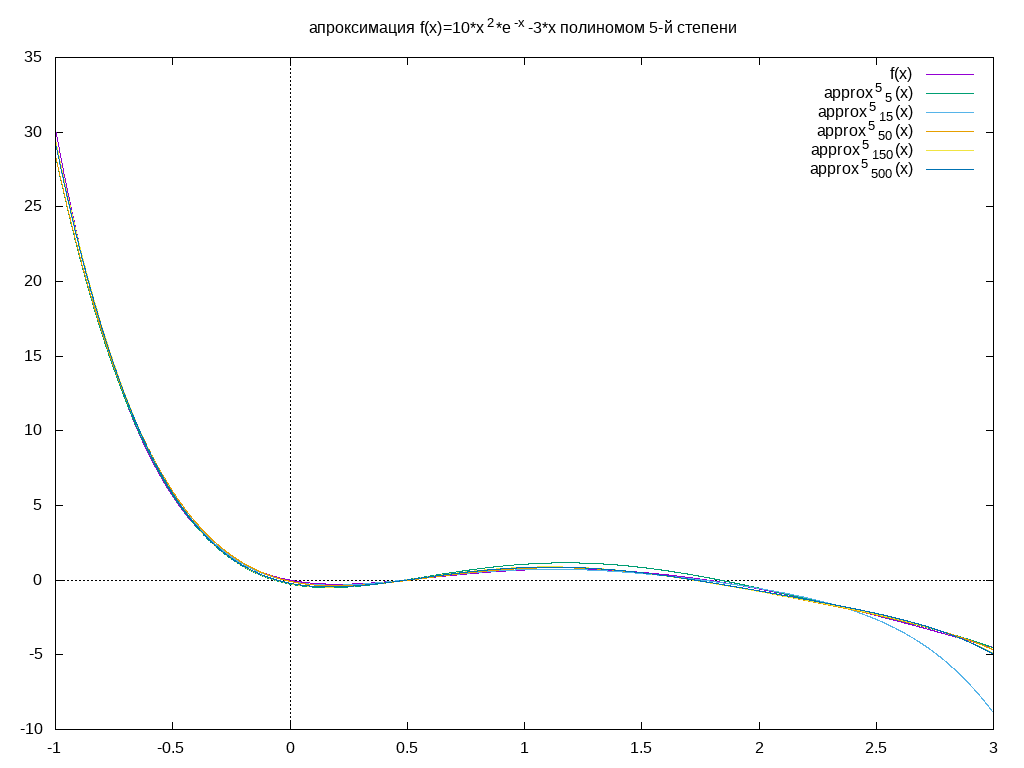
make(5\*100, 5);

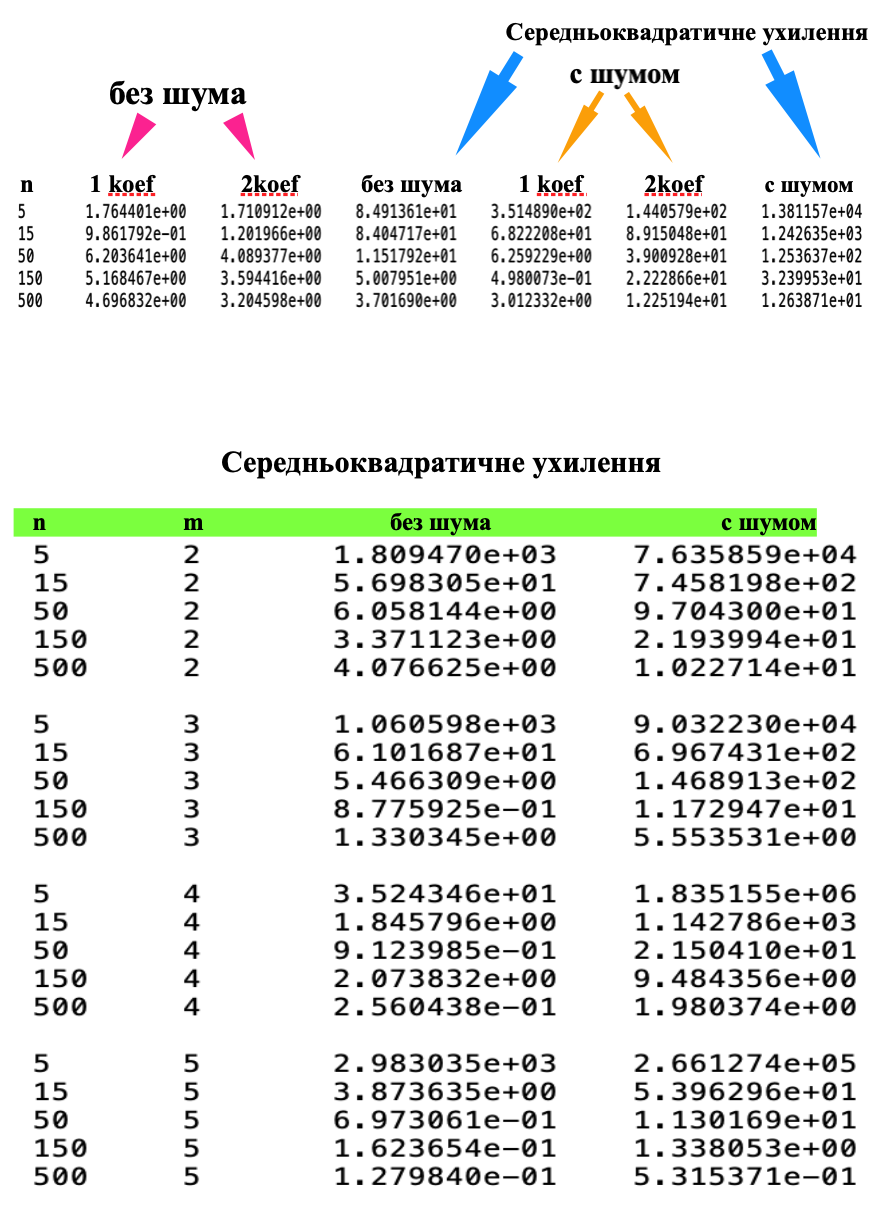
return 0;

}

**Результати**

****

****

****

**Висновок:** я дослідив метод найменших квадратів, суть якого полягає в тому, що сукупність параметрів с={c1,c2,...cm} визначається вимогою мінімізації суми квадратів похибок кожного експерименту і виявив що Середньоквадратичнe ухилення при збільшенні вузлів прямує до нуля, а також цей процес спостерігаеться при збільшенні m. Побудував наближення функції f(x) лінійним поліномом за методом найменших квадратів, використовуючи набір із n псевдовипадкових «експериментальних» точок (xі,yі)(i=1,2,...n), що розташовані на кривій f(x) та імітують серію вимірювань. Додатково – побудувати МНК-наближення поліномами більш високих степенів та дослідити залежність σвід кількості точокnта степеніМНК-полінома m.

Всі графіки були побудовані за допомогою програма для створення дво- і тривимірних графіків **gnuplot.**