

СОДЕРЖАНИЕ.

[**1.Постановка задачи** 3](#_Toc90084842)

[**2.Представление исходных данных(табличное)** 3](#_Toc90084843)

[**3. Описание критерия аппроксимации и способа его минимизации** 3](#_Toc90084844)

[**4.Описание заданного метода вычисления коэффициентов нормальных уравнений.** 4](#_Toc90084845)

[**5. Расчет параметров аппроксимирующей функции (ручной счет).** 6](#_Toc90084846)

[**6.Схемы алгоритмов и их описание** 11](#_Toc90084847)

[**7.Текст программы и результат расчёта на ЭВМ.** 21](#_Toc90084848)

[**8.Графическое сопоставление исходной и аппроксимирующей (ϕ\*(x) и ϕ\*\*(x)) функциональных зависимостей.** 26](#_Toc90084849)

[**9.Оценка погрешности аппроксимации.** 29](#_Toc90084850)

[**10. Выводы.** 30](#_Toc90084851)

## **1.Постановка задачи**

Исходными данными являются функциональная зависимость между X и Y, заданная таблично в n точках, и набор базисных функций. Требуется, используя МНК(метод наименьших квадратов), найти аппроксимирующую функцию ϕ(x) из заданного класса функций и оценить степень ее отклонения от исходной функции.

## **2.Представление исходных данных(табличное)**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер  варианта | n | Значения xi и yi , i = 1, …, n ϕ1(x) | | | | | | Базисные функции | | | Метод решения СЛАУ |
| ϕ1(x) | ϕ2(x) | ϕ3(x) |
| 19 | 5 | X | 0,0 | 0,78 | 1,57 | 2,35 | 3,14 | cos(x) | sin(x) | 1 | Обратная матрица |
| Y | 0,0 | 4,0 | 5,0 | 8,0 | 9,0 |

## **3. Описание критерия аппроксимации и способа его минимизации**

При изучении зависимостей между величинами важной задачей является приближённое представление (Аппроксимация) этих зависимостей с помощью известных функций или их комбинаций, подобранных надлежащим образом. Подход к такой задаче и конкретный метод её решения определяются выбором используемого критерия качества приближения и формой представления исходных данных.

Пусть изучается связь между величинами X и Y, из которых первая рассматривается в качестве независимой переменной, а вторая её функции. Исходные данные представлены значениями Y, заданными на некотором множестве M функции X. Тогда ошибка приближения этой зависимости некоторой аппроксимирующей функции y = f(x) для каждого из значений X может быть оценена разностью g = y – f(x) x∈M.

Метод наименьших квадратов говорит о том, что наилучшей считается такая аппроксимирующая функция f(x), для которой достигается наименьшее значение суммы квадратов ошибок во всех точках x, принимаемых во внимание. Это требование имеет вид:

Значит наилучшая аппроксимирующая функция должна быть определена из условий:

## **4.Описание заданного метода вычисления коэффициентов нормальных уравнений.**

Один из методов решения системы линейных алгебраических уравнений, записываемой в матричной форме A × C = B, связан с использованием обратной матрицы A-1.

В этом случае решение системы уравнений получается в виде:

C = А-1 × B,

где А-1 – матрица, определяемая следующим образом.

Пусть А – квадратная матрица размером m\*m c ненулевым определителем det A ≠ 0. Тогда существует обратная матрица R = А-1, определяемая условием :

A \* R = E,

где Е – единичная матрица, размерности m\*m все элементы главной диагонали которой равны 1, а все элементы вне этой диагонали равны 0:

E=

Матрица R – квадратная матрица размером m\*m:

R=

Тогда можно записать в матричном виде:

Умножим матрицу А на первый столбец матрицы R . Получим первый столбец матрицы Е, то есть система уравнений будет представлена в виде:

a11 r11 + a12 r12 +…+ a1m rm1 = 1

a21 r11 + a22 r21 +…+ a2m rm1 = 0

……………………………………

am1 r11 + am2 r12 +…+ amm rm1 = 0

где неизвестными являются значения r11, r21, …, rm1. Решением этой системы будет набор значений r11, r21, …, rm1, то есть первый столбец матрицы R.

Аналогично, умножая матрицу A на второй столбец матрицы R, получим:

a11 r12 + a12 r22 +…+ a1m rm2 = 1

a21 r12 + a22 r22 +…+ a2m rm2 = 0

……………………………………

am1 r12 + am2 r22 +…+ amm rm2 = 0

Решением этой системы будет набор значений r12, r22, …, rm2, то есть второй столбец матрицы R. И так далее до m–го столбца матрицы R:

a11 r1m + a12 r2m +…+ a1m rmm = 1

a21 r1m + a22 r2m +…+ a2m rmm = 0

……………………………………

am1 r1m + am2 r2m +…+ amm rmm = 0

Решением этой системы будет набор значений r1m, r2m, …, rmm, то есть m– й столбец матрицы R. В результате получим m систем уравнений. Для решения этих систем можно применять любые методы, разработанные для решения систем линейных алгебраических уравнений. вычислив матрицу R (матрицу A-1), находим матрицу коэффициентов С в соответствии с C = А-1 × B.

## **5. Расчет параметров аппроксимирующей функции (ручной счет).**

1. Выражение для аппроксимирующей функции будет иметь следующий вид:
2. Запишем выражения для критерия аппроксимации:
3. В соответствии с условиями локального минимума функции J(C1, C2, C3) найдем частные производные , , и приравняем их к нулю:

Таким образом получим три уравнения:

1. Приведем полученную систему уравнений к нормальному виду , перенеся свободные члены вправо и поделив обе части уравнений на 2.
2. Для удобства представим промежуточные результаты вычислений в виде таблицы.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n | x | y | cos(x)^2 | cos(x) | y\*cos(x) | cos(x)\*sin(x) | sin(x)^2 | sin(x) | y\*sin(x) |
| 1 | 0,000000 | 0,000000 | 1,000000 | 1,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 | 0,000000 |
| 2 | 0,780000 | 4,000000 | 0,505398 | 0,710914 | 2,843654 | 0,499971 | 0,494602 | 0,703279 | 2,813118 |
| 3 | 1,570000 | 5,000000 | 0,000001 | 0,000796 | 0,003982 | 0,000796 | 0,999999 | 1,000000 | 4,999998 |
| 4 | 2,350000 | 8,000000 | 0,493806 | -0,702713 | -5,621705 | -0,499962 | 0,506194 | 0,711473 | 5,691787 |
| 5 | 3,140000 | 9,000000 | 0,999997 | -0,999999 | -8,999989 | -0,001593 | 0,000003 | 0,001593 | 0,014334 |
| ∑ | 7,840000 | 26,000000 | 2,999202 | 0,008998 | -11,774057 | -0,000787 | 2,000798 | 2,416345 | 13,519237 |

Таблица 1.Промежуточные результаты

Используя значения из таблицы запишем систему уравнений в окончательном виде:

1. Обозначим через А — матрицу коэффициентов при неизвестных; X — матрицу-столбец неизвестных; B - матрицу-столбец свободных членов:

|  |  |  |
| --- | --- | --- |
| 2,999202 | -0,000787 | 0,008998 |
| -0,000787 | 2,000797 | 2,416345 |
| 0,008998 | 2,416345 | 5 |

Вектор B:  
BT=(-11.774057,13.519237,26)  
С учетом этих обозначений данная система уравнений принимает следующую матричную форму: А\*Х = B.  
Если матрица А — невырожденная (ее определитель отличен от нуля, то она имеет обратную матрицу А-1. Умножив обе части уравнения на А-1, получим: А-1\*А\*Х = А-1\*B, А-1\*А=Е.  
Это равенство называется матричной записью решения системы линейных уравнений. Для нахождения решения системы уравнений необходимо вычислить обратную матрицу А-1.  
Система будет иметь решение, если определитель матрицы A отличен от нуля.  
Найдем главный определитель:

Итак, определитель 12.492262332258 ≠ 0, поэтому продолжаем решение. Для этого найдем обратную матрицу через алгебраические дополнения.  
Пусть имеем невырожденную матрицу А:

|  |  |  |
| --- | --- | --- |
| a11 | a21 | a31 |
| a12 | a22 | a32 |
| a13 | a23 | a33 |

A=

|  |  |  |
| --- | --- | --- |
| A11 | A21 | A31 |
| A12 | A22 | A32 |
| A13 | A23 | A33 |

Тогда:

Где Aij — алгебраическое дополнение элемента aij в определителе матрицы А, которое является произведением (—1)i+j на минор (определитель) n-1 порядка, полученный вычеркиванием i-й строки и j-го столбца в определителе матрицы А.

Транспонированная матрица к матрице A имеет вид:

|  |  |  |
| --- | --- | --- |
| 2,999202 | -0,000787 | 0,008998 |
| -0,000787 | 2,000797 | 2,416345 |
| 0,008998 | 2,416345 | 5 |

=

Вычислим алгебраические дополнения:

Из полученных алгебраических дополнений составим присоединенную матрицу C:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

C=

Вычислим обратную матрицу:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

=

Вектор результатов X

X=A-1 • B

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

|  |
| --- |
| -49,2124418913638158105 |
| 14,006233214012993382 |
| 58,279548477715607005 |

C1=3,9394339137664085567770373165705

C2=1,1211926904421114799773590952289

C3=4,6652517316438285678354174534716

1. В результате решения исходной системы линейных уравнений и нахождения значений C1\*, С2\*, С3\* получаем запись искомой аппроксимирующей функции в следующем виде:
2. Оценка погрешности аппроксимации.

Таблица 2.Оценка погрешности аппроксимации.

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | F(x) | Дельта |
| 0,00 | 0,00 | 0,725817817877420 | 0,725817817877420 |
| 0,78 | 4,00 | 2,653166574388470 | 1,346833425611530 |
| 1,57 | 5,00 | 5,783306990140720 | 0,783306990140719 |
| 2,35 | 8,00 | 8,231242180526130 | 0,231242180526126 |
| 3,14 | 9,00 | 8,606466319943050 | 0,393533680056947 |

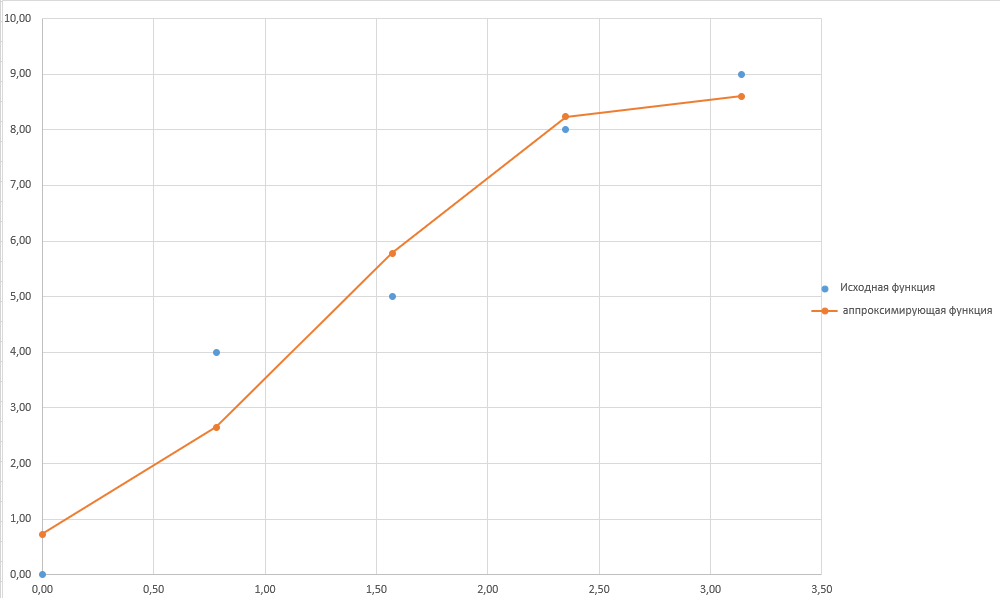


Рисунок 1. График исходной и аппроксимируемой функции.

Вычислим значение критерия аппроксимации, подставив данные из таблицы:

Максимальное по модулю отклонение =1,346833425611530, при x2=0,78.

## **6.Схемы алгоритмов и их описание**

Представим блок-схемы программы и используемых в ней функций:

Функция AklF:

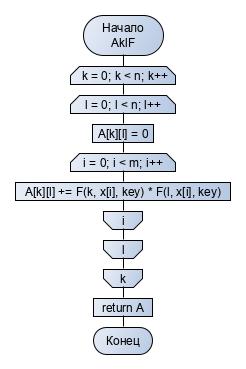


Рисунок 2 - Блок-схема функции AklF

Вычисляет коэффициенты a при неизвестных в матрицу A.

Функция BkF:

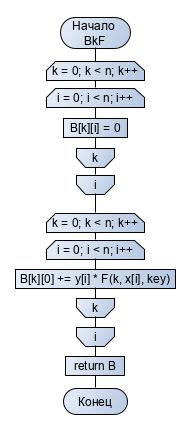


Рисунок 3 - Блок-схема функции BkF

Вычисляет свободные коэффициенты b в вектор B.

Функция Determinant:

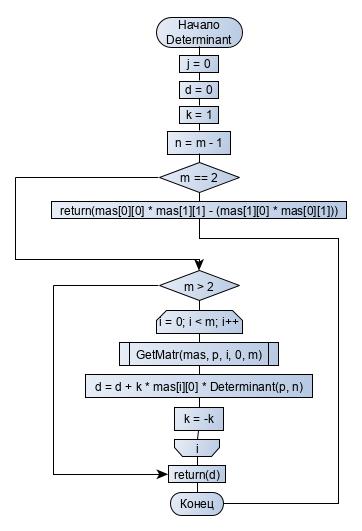


Рисунок 4 - Блок-схема функции Determinant

Рекурсивно вычисляет детерминант.

Функция Enter:

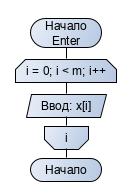


Рисунок 5 - Блок-схема функции Enter

Ввод исходных значений в массив.

Функция F:

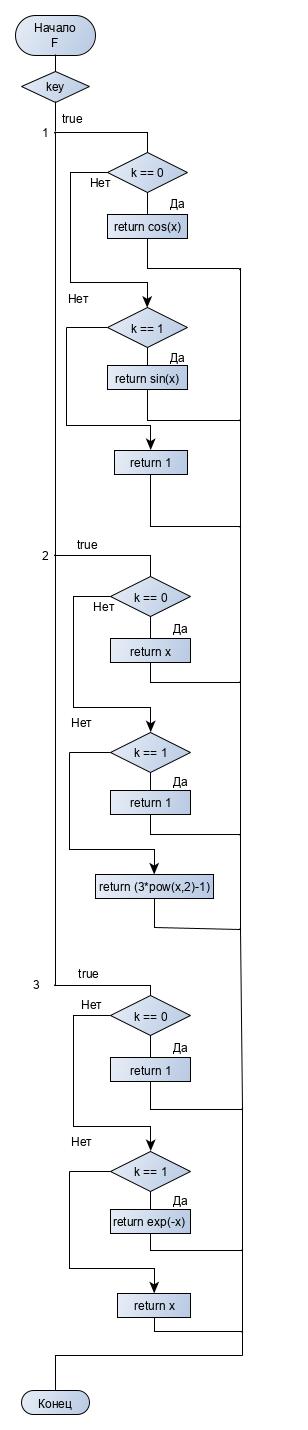


Рисунок 6 - Блок-схема функции F

Выбирает набор функций в зависимости от значения переменной key, а после возвращает значение функции в зависимости от этапа суммирования при нахождение коэффициентов.

Функция GetMatr:

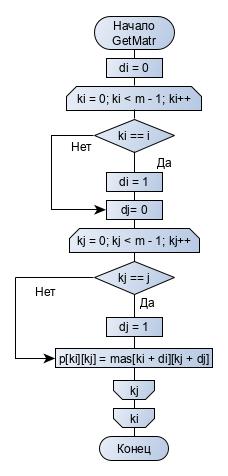


Рисунок 7 - Блок-схема функции GetMatr

Убирает I-тый столбец j-тую строку из матрицы.

Функция InvMatr:

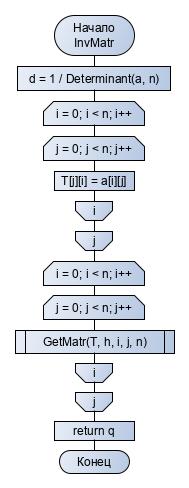


Рисунок 8 - Блок-схема функции InvMatr

Вычисляет обратную матрицу и возвращает указатель на её первый элемент.

Функция ProdMatr:

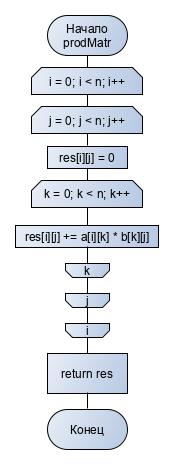
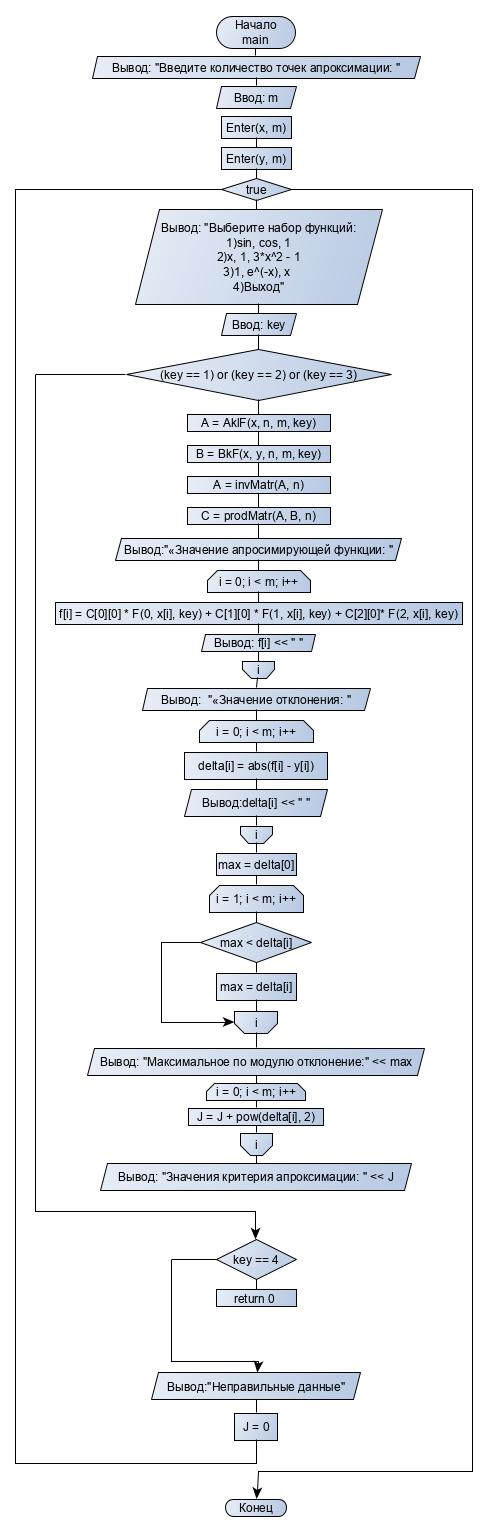


Рисунок 9 - Блок-схема функции ProdMatr

Умножает две матрицы и возвращает указатель на результирующую матрицу.

Главная функция main:

  
*Рисунок 10 - Блок-схема функции main*

Вводятся значения x и y, открывается цикл, позволяющий произвести несколько расчётов, выбирается набор функций вводом переменной key, если она равна 1, 2 или 3, происходит вычисления матрицы А и вектора B, находится обратная матрица A-1, умножается матрица A на B, результат записывается в матрицу C, после чего высчитываются и выводятся значения аппроксимирующей функции, значения отклонения, выбирается максимальное отклонение, и, в конце, считается значение критерия аппроксимации. Если переменная key = 4, то происходит выход из программы.

## **7.Текст программы и результат расчёта на ЭВМ.**

Текст программы:

#include <iostream>

#include <cmath>

using namespace std;

double F(int k, double x, int key);

void GetMatr(double\*\* mas, double\*\* p, int i, int j, int m);

double Determinant(double\*\* mas, int m);

void print(double\*\* A, int n);

//Ввод

void Enter(double\* x, int m) {

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

cin >> x[i];

}

//получение матрицы без i-той строки и j-того столбца

void GetMatr(double\*\* mas, double\*\* p, int i, int j, int m) {

int ki, kj, di = 0, dj;

for (ki = 0; ki < m - 1; ki++) { // проверка индекса строки

if (ki == i) di = 1;

dj = 0;

for (kj = 0; kj < m - 1; kj++) { // проверка индекса столбца

if (kj == j) dj = 1;

p[ki][kj] = mas[ki + di][kj + dj];

}

}

}

//нахождение детерминанта

double Determinant(double\*\* mas, int m) {

int i, j, k, n;

double\*\* p, d;

p = new double\* [m];

for (i = 0; i < m; i++)

p[i] = new double[m];

j = 0; d = 0;

k = 1; //(-1) в степени i

n = m - 1;

if (m == 2)

return(mas[0][0] \* mas[1][1] - (mas[1][0] \* mas[0][1]));

if (m > 2) {

for (i = 0; i < m; i++) {

GetMatr(mas, p, i, 0, m);

d = d + k \* mas[i][0] \* Determinant(p, n);

k = -k;

}

}

return(d);

}

//нахождение обратной матрицы

double\*\* invMatr(double\*\* a, int n) {

double\*\* q = new double\* [n];

double\*\* T = new double\* [n];

double\*\* h = new double\* [n];

double d;

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

q[i] = new double[n];

T[i] = new double[n];

h[i] = new double[n];

}

d = 1 / Determinant(a, n);

//Транспонирование

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

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

T[j][i] = a[i][j];

}

//Матрица алгебраических дополнений

cout << endl;

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

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

GetMatr(T, h, i, j, n);

q[i][j] = d \* Determinant(h, 2);

if ((i + j + 2) % 2 == 1)

q[i][j] = q[i][j] \* (-1);

}

return q;

}

//Умножение матриц

double\*\* prodMatr(double\*\* a, double\*\* b, int n)

{

double\*\* res;

double sum = 0;

res = new double\* [n];

int i, j;

res = new double\* [n];

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

res[i] = new double[n];

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

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

res[i][j] = 0;

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

res[i][j] += a[i][k] \* b[k][j];

}

}

return res;

}

//Расчёт коэффициентов

double\*\* AklF(double\* x, int n, int m, int key) {

double\*\* A = new double\* [n];

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

A[i] = new double[n];

for (int k = 0; k < n; k++) {// 3 - количество базисных функций

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

A[k][l] = 0;

for (int i = 0; i < m; i++) { // 5 - количество точек аппроксимации

A[k][l] += F(k, x[i], key) \* F(l, x[i], key);

}

}

}

return A;

}

double\*\* BkF(double\* x, double\* y, int n, int m, int key) {

double\*\* B = new double\* [n];

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

B[i] = new double[n];

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

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

B[k][i] = 0;

}

}

for (int k = 0; k < n; k++) {// 3 - количество базисных функций

for (int i = 0; i < m; i++) { // 5 - количество точек апроксимации

B[k][0] += y[i] \* F(k, x[i], key);

}

}

return B;

}

double F(int k, double x, int key) {

switch (key)

{

case (1):

if (k == 0)

return cos(x);

if (k == 1)

return sin(x);

else

return 1;

break;

case (2):

if (k == 0)

return 1;

if (k == 1)

return (x);

else

return (3\*pow(x,2)-1);

break;

case (3):

if (k == 0)

return 1;

if (k == 1)

return exp(-x);

else

return x;

break;

}

}

int main() {

setlocale(LC\_ALL, "Russian");

int n = 3, m, key;

double J = 0, max = 0;

cout << "Введите количество точек апроксимации: ";

cin >> m;

double\*\* A;

double\*\* B;

double\*\* C;

double\* x = new double[m];

double\* y = new double[m];

double\* f = new double[m];

double\* delta = new double[m];

Enter(x, m);

Enter(y, m);

while (true) {

cout << "Выберите набор функций:" << endl

<< "1)sin, cos, 1 " << endl

<< "2)x, 1, 3\*x^2 - 1" << endl

<< "3)1, e^(-x), x " << endl

<< "4)Выход" << endl;

cin >> key;

if ((key == 1) or (key == 2) or (key == 3)) {

A = AklF(x, n, m, key);

B = BkF(x, y, n, m, key);

A = invMatr(A, n);

C = prodMatr(A, B, n);

cout << "Значения апросимирующей функции: ";

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

f[i] = C[0][0] \* F(0, x[i], key) + C[1][0] \* F(1, x[i], key) + C[2][0]\* F(2, x[i], key);

cout << f[i] << " ";

}

cout << endl << "Значения отклонения: ";

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

delta[i] = abs(f[i] - y[i]);

cout << delta[i] << " ";

}

max = delta[0];

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

if (max < delta[i])

max = delta[i];

}

cout << endl << "Максимальное по модулю отклонение:" << max;

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

J = J + pow(delta[i], 2);

cout << endl << "Значения критерия аппроксимации: " << J << endl;

}

else if (key == 4)

return 0;

else

cout << "Неправильные данные" << endl;

J = 0;

system("pause");

system("cls");

}

}

Результат выполнения программы:

Первый набор функций:

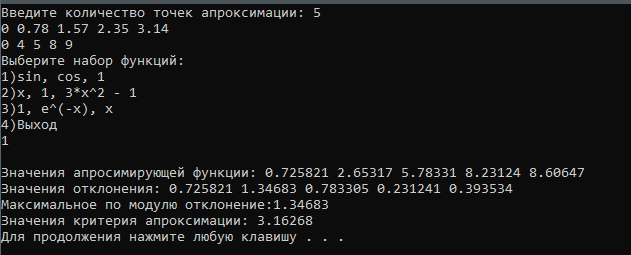


Рисунок 11 - результат выполнения программы в первом случае

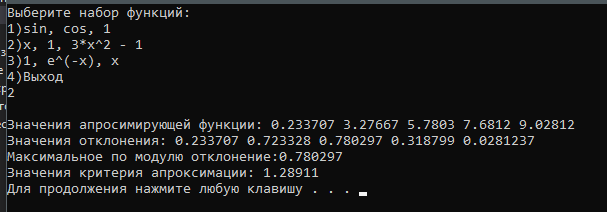
Второй набор функций:  


Рисунок 12 - результат выполнения программы во втором случае

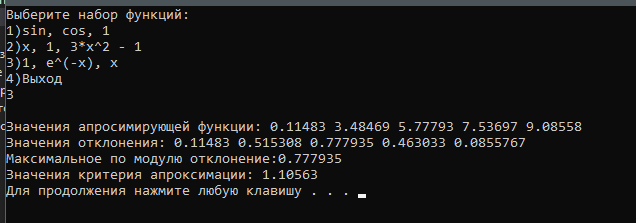
Третий набор функций:  


Рисунок 13 - результат выполнения программы в третьем случае

## **Графическое сопоставление исходной и аппроксимирующей (ϕ\*(x) и ϕ\*\*(x)) функциональных зависимостей.**

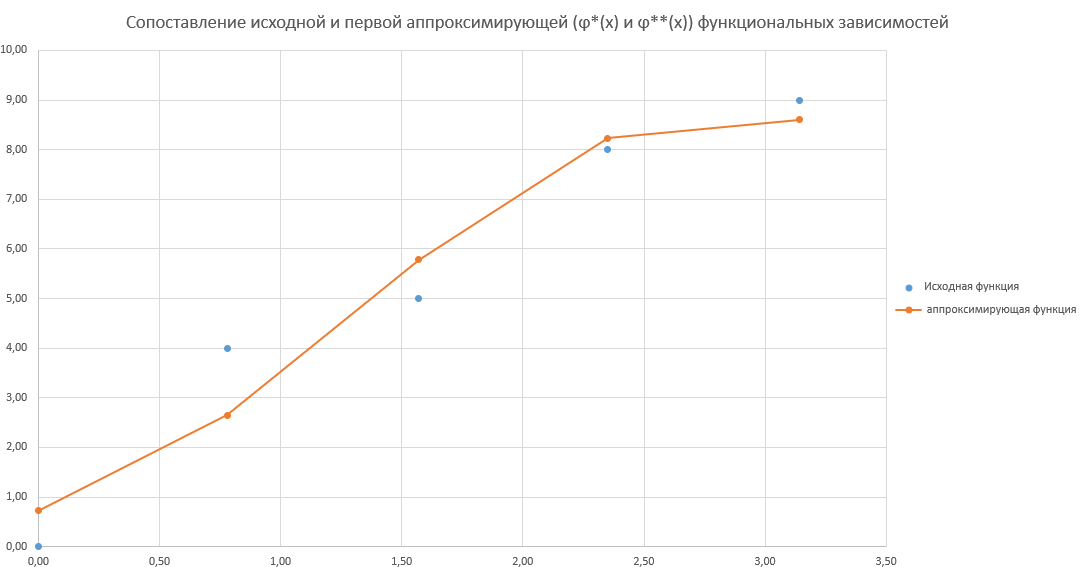
Сопоставление исходной и первой аппроксимирующей функциональной зависимостей:

Рисунок 14 – график исходной и первой аппроксимирующей функциональной зависимостей

Сопоставление исходной и второй аппроксимирующей функциональной зависимостей.

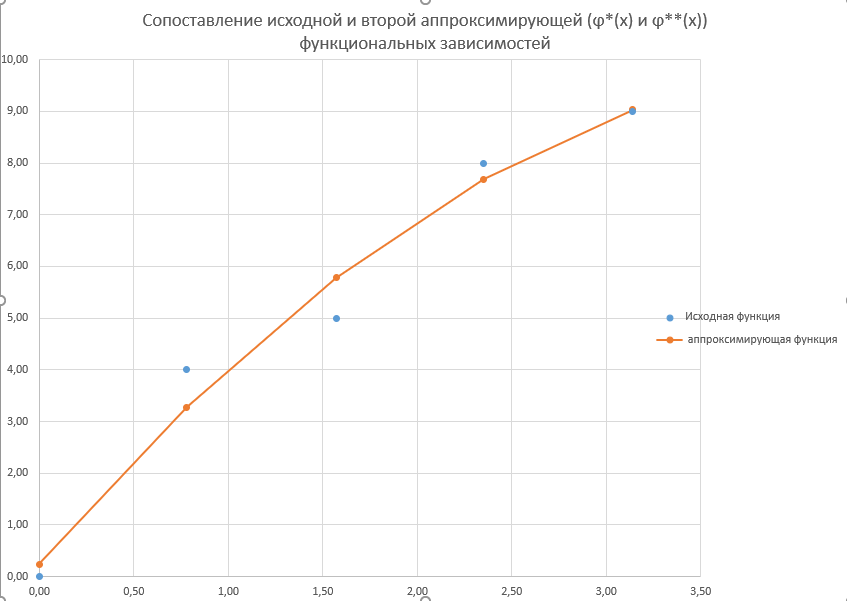


Рисунок 15 - график исходной и второй аппроксимирующей функциональной зависимостей

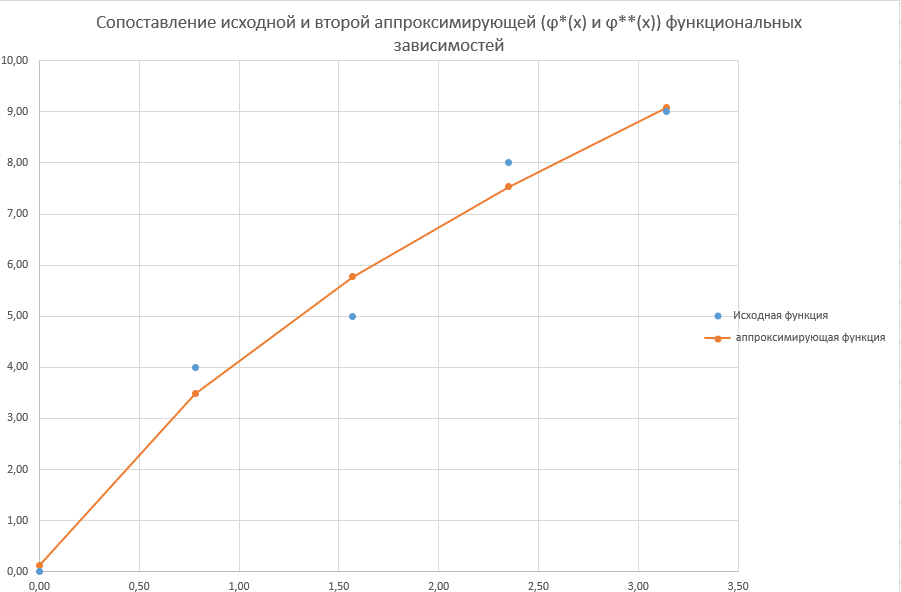
Сопоставление исходной и третьей аппроксимирующей функциональной зависимостей:

Рисунок 16 - график исходной и третьей аппроксимирующей функциональной зависимостей

## **Оценка погрешности аппроксимации.**

С помощью программы были вычислены 3 критерия аппроксимации по 3 наборам функций  
J1 = 3.16268

J2 = 1.28911

J3 = 1.10563

Критерий по исходному набору функций(первый) совпал со значением, посчитанным вручную.

Максимальный модуль погрешности равен:

Максимальный модуль отклонения получен в третьем наборе функций. Минимальный критерий аппроксимации получен в третьем наборе функций.

## **10. Выводы.**

В процессе выполнения курсовой работы на практике были освоены методы вычислительной математике применяемые для аппроксимации функции, а также улучшены навыки разработки алгоритмов и построения программы на высокоуровневом языке С++. Полученные навыки являются основой вычислительных методов прикладной математики и техники программирования во всех последующих дисциплинах и будут полезны при выполнение курсовых и дипломных проектов.

Значения критерия аппроксимации ручного и машинного счёта полностью совпадают вплоть до 5 знаков после запятой, что является достаточной точностью.

Различие в массивах коэффициентов, а следовательно, и на графиках, аппроксимирующих функции вычисленных обоими способами, также отсутствуют. Для вычисления более точного критерия аппроксимации были выбраны ещё два набора функций. Наилучший критерий аппроксимации был получен с помощью третьего набора функций (1, e^(-x), x), J3 = 1.10563. В итоге максимально приближенная аппроксимирующая функция выглядит так:

.