МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»

РУТ (МИИТ)

Кафедра ЦТУТП

**ЛАБОРАТОРНАЯ РАБОТА №3**

ПО ДИСЦИПЛИНЕ «ЧИСЛЕННЫЕ МЕТОДЫ»

НА ТЕМУ «ДИСКРЕТНОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ И ТРИГОНОМЕТРИЧЕСКИЙ ИНТЕРПОЛЯЦИОННЫЙ МНОГОЧЛЕН»

ВАРИАНТ №6

Выполнил: студент группы

УПМ-311 Масеёнок Е.Н.

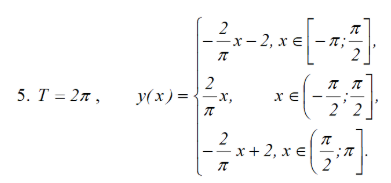
Проверил: Посвянский В.П.

МОСКВА 2022

**Постановка задачи**

1. Составить таблицы значений
2. Определить величину
3. Протабулировать функцию и многочлен на отрезке с шагом
4. Для каждого *n* начертить на одном рисунке график функции и многочлена.

Исходная функция



**Код программы**

#define \_USE\_MATH\_DEFINES

#include <iostream>

#include <cmath>

#include <iomanip>

#include<fstream>

using namespace std;

double function(const double x)

{

double fx = 0;

if (x <= -M\_PI/2)

{

fx = (2 / M\_PI) \* (x + M\_PI);

}

else {

fx = ( -((2 \* x) / (3 \* M\_PI)) + (2.0 / 3));

}

return fx;

}

double\* fillXn(const int n , const double T)

{

double h = T / (2 \* n);

double\* x = new double[2 \* n]{ -T / 2 + h };

for (int i = 1; i < 2 \* n; i++)

{

x[i] = x[i - 1] + h;

}

return x;

}

double\* fillYn(const int n,const double x[], const double T)

{

double h = T / (2 \* n);

double\* y = new double[2 \* n]{ -T / 2 + h };

for (int i = 0; i < 2 \* n; i++)

{

y[i] = function(x[i]);

}

return y;

}

double findAlpha0(const int n , double y[])

{

double a0 = 0;

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

a0 = a0 + y[k + n - 1];

}

a0 = a0 / (2 \* n);

return a0;

}

double\* findAlphaArray(const int n, const double y[],const double T)

{

double\* al = new double[n];

double h = T / (2 \* n); //2nh=T формула

double w = 2 \* M\_PI / T;

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

{

double temp = 0;

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

{

temp = temp + y[k + n - 1] \* cos((l + 1) \* w \* k \* h);

}

if (l == n - 1)

al[l] = (h \* temp) / T;

else

al[l] = (h \* temp \* 2) / T;

}

return al;

}

double\* findBetaArray(const int n, const double y[], const double T)

{

double\* bl = new double[n] {};

double h = T / (2 \* n); //2nh=T формула

double w = 2 \* M\_PI / T;

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

{

double temp = 0;

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

{

temp = temp + y[k + n - 1] \* sin((l + 1) \* w \* k \* h);

}

if (l == n - 1)

bl[l] = (h \* temp) / T;

else

bl[l] = (h \* temp \* 2) / T;

}

return bl;

}

void printXandY(const int n , const double x[] , const double y[])

{

cout << "n = "<<n<<" : " << endl;

if (n < 10)

{

cout << "Xk | ";

for (int i = 0; i < 2 \* n; i++)

{

cout << setw(7) << fixed << setprecision(3) << x[i] << " | ";

}

cout << endl;

cout << "yk | ";

for (int i = 0; i < 2 \* n; i++)

{

cout << setw(7) << fixed << setprecision(3) << y[i] << " | ";

}

}

else

{

cout << "Xk | ";

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

{

cout << setw(7) << fixed << setprecision(3) << x[i] << " | ";

}

cout << endl;

cout << "yk | ";

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

{

cout << setw(7) << fixed << setprecision(3) << y[i] << " | ";

}

cout << endl;

cout << "Xk | ";

for (int i = n; i < 2\*n; i++)

{

cout << setw(7) << fixed << setprecision(3) << x[i] << " | ";

}

cout << endl;

cout << "yk | ";

for (int i = n; i < 2\*n; i++)

{

cout << setw(7) << fixed << setprecision(3) << y[i] << " | ";

}

}

cout << endl <<endl<<endl;

}

void printALPHAandBETA(const int n,const double alpha0, const double al[], const double bl[])

{

cout << "n = " << n << " , " << " a0 = " << alpha0 << " : " << endl;;

cout << "al | ";

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

{

cout << setw(7) << fixed << setprecision(3) << al[i] << " | ";

}

cout << endl;

cout << "bl | ";

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

{

cout << setw(7) << fixed << setprecision(3) << bl[i] << " | ";

}

cout << endl << endl << endl;

}

double\* Fourier\_interpolation(const int n, int iters, const double alpha0, const double\* al, const double\* bl, const double x[],const double T)

{

double h = T / (2 \* n); //2nh=T формула

double w = 2 \* M\_PI / T;

double\* y = new double[iters] {};

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

{

double temp = alpha0;

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

{

temp = temp + al[l] \* cos((l + 1) \* w \* x[i]) + bl[l] \* sin((l + 1) \* w \* x[i]);

}

y[i] = temp;

}

return y;

}

double\* findX(const int iters, const double T)

{

double\* x = new double[iters ]{-T/2};

double h = T / iters;

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

{

x[i] = x[i - 1] + h;

}

return x;

}

double\* findY(const int iters, const double x[])

{

double\* y = new double[iters ] {};

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

{

y[i] = function(x[i]);

}

return y;

}

void printFinal(const int iters,const double x[], const double y[], const double fourier\_n4[], const double fourier\_n8[], const double fourier\_n16[])

{

cout << setw(5) << fixed << setprecision(3) << "x ";

cout << setw(5) << fixed << setprecision(3) << "y(x) ";

cout << setw(5) << fixed << setprecision(3) << "f(x) ";

cout << setw(5) << fixed << setprecision(3) << "f(x) ";

cout << setw(5) << fixed << setprecision(3) << "f(x) ";

cout << endl;

cout << setw(5) << fixed << setprecision(3) << " ";

cout << setw(5) << fixed << setprecision(3) << " ";

cout << setw(5) << fixed << setprecision(3) << "n=4 ";

cout << setw(5) << fixed << setprecision(3) << "n=8 ";

cout << setw(5) << fixed << setprecision(3) << "n=16 ";

cout << endl;

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

{

cout << setw(7) << fixed << setprecision(3) << x[i] << " ";

cout << setw(7) << fixed << setprecision(3) << y[i] << " ";

cout << setw(7) << fixed << setprecision(3) << fourier\_n4[i] << " ";

cout << setw(7) << fixed << setprecision(3) << fourier\_n8[i] << " ";

cout << setw(7) << fixed << setprecision(3) << fourier\_n16[i] << " ";

cout << endl;

}

}

int main()

{

const double T = 2\*M\_PI; // период функции

double\* x4 = fillXn(4, T);

double\* y4 = fillYn(4, x4, T);

printXandY(4, x4, y4);

double\* x8 = fillXn(8, T);

double\* y8 = fillYn(8, x8, T);

printXandY(8, x8, y8);

double\* x16 = fillXn(16, T);

double\* y16 = fillYn(16, x16, T);

printXandY(16, x16, y16);

double alpha0n4 = findAlpha0(4, y4);

double\* aln4 = findAlphaArray(4, y4, T);

double\* bln4 = findBetaArray(4, y4, T);

printALPHAandBETA(4, alpha0n4, aln4, bln4);

double alpha0n8 = findAlpha0(8, y8);

double\* aln8 = findAlphaArray(8, y8, T);

double\* bln8 = findBetaArray(8, y8, T);

printALPHAandBETA(8, alpha0n8, aln8, bln8);

double alpha0n16 = findAlpha0(16, y16);

double\* aln16 = findAlphaArray(16, y16, T);

double\* bln16 = findBetaArray(16, y16, T);

printALPHAandBETA(16, alpha0n16, aln16, bln16);

double\* x50 = findX(50, T);

double\* y50 = findY(50,x50);

double\* fn4 = Fourier\_interpolation(4, 50, alpha0n4, aln4, bln4, x50, T);

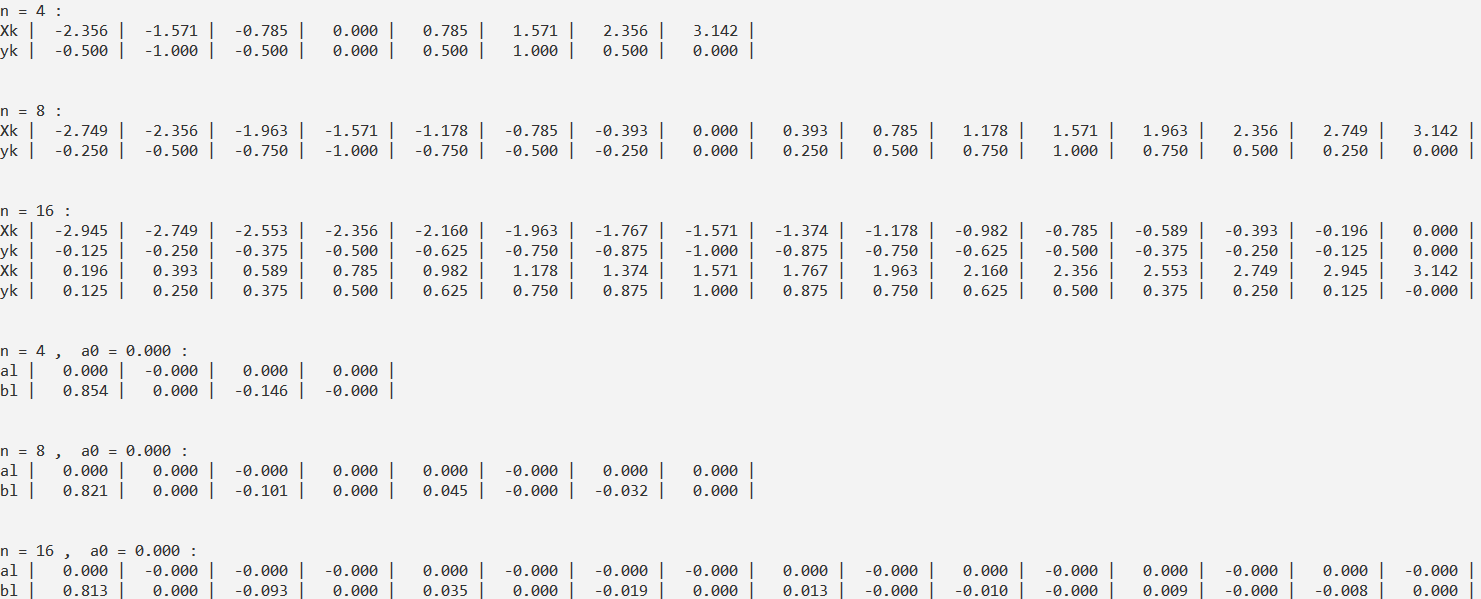
double\* fn8 = Fourier\_interpolation(8, 50, alpha0n8, aln8, bln8, x50, T);

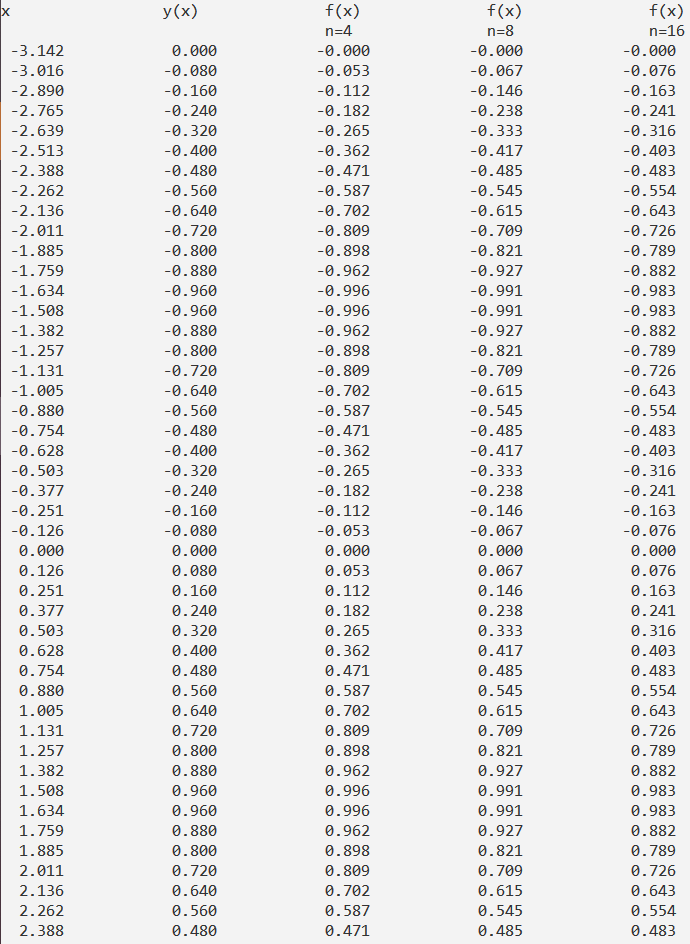
double\* fn16 = Fourier\_interpolation(16, 50, alpha0n16, aln16, bln16, x50, T);

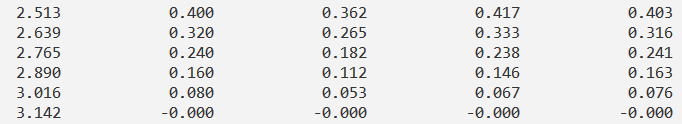
printFinal(50, x50, y50, fn4,fn8,fn16);

}

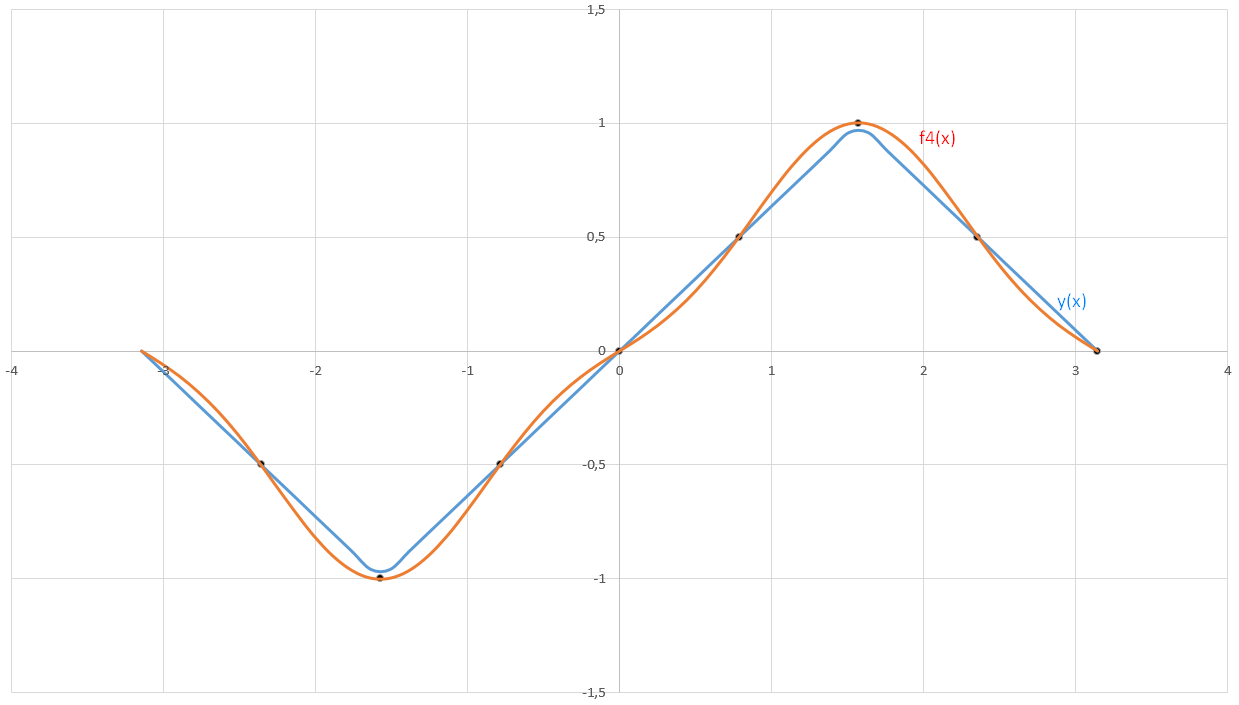
**Результаты работы программы**

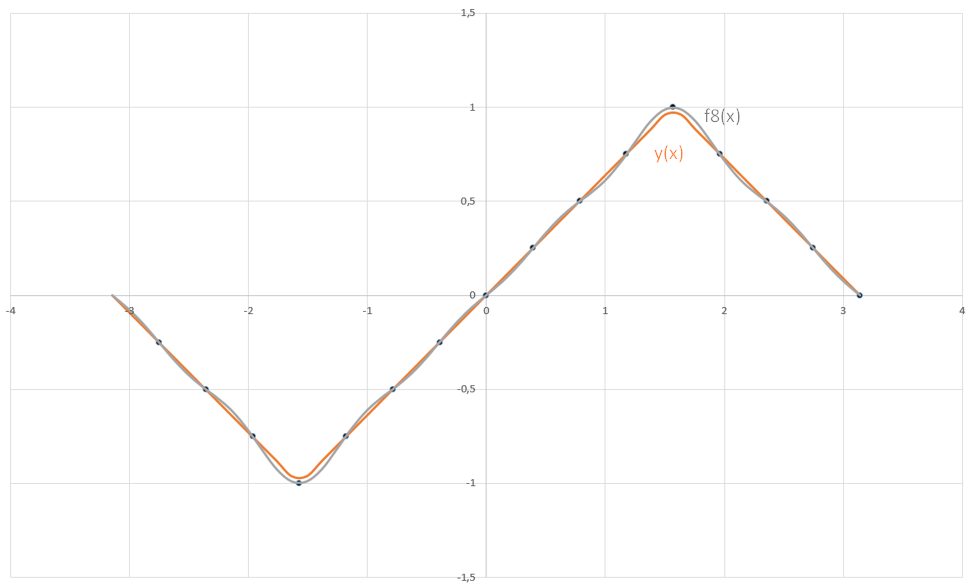


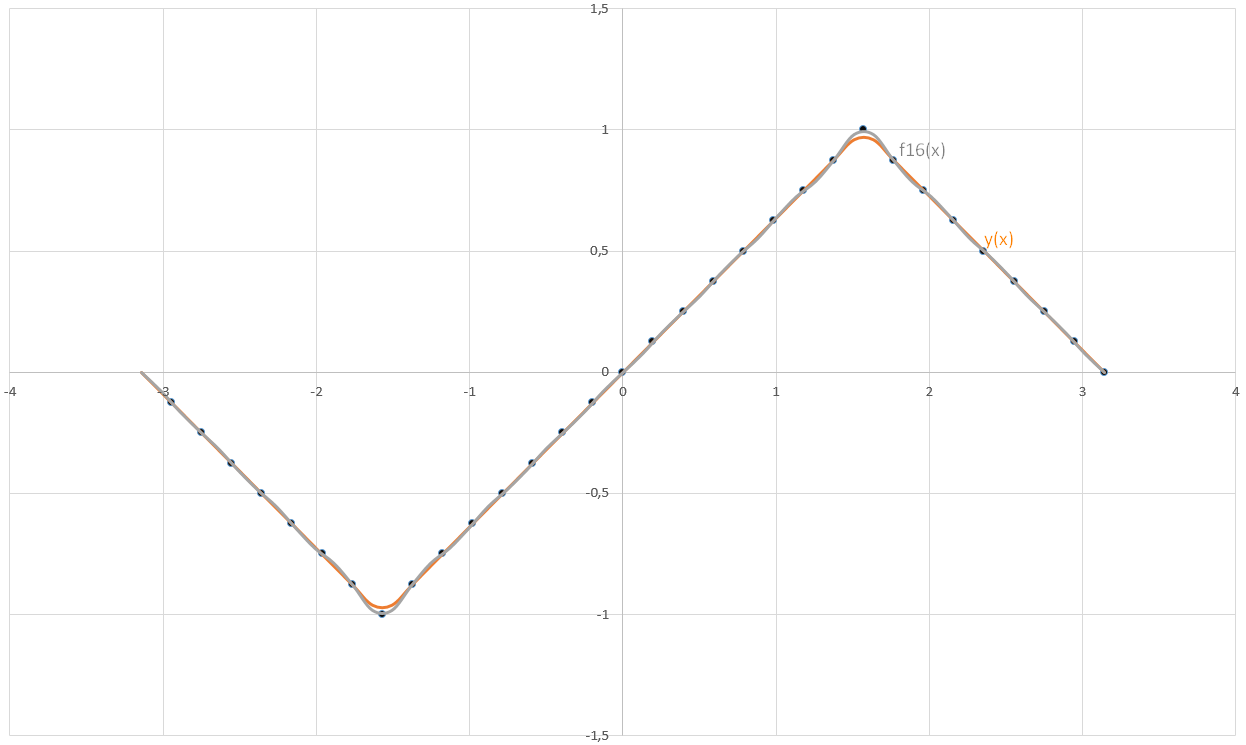




Графики функций







**Вывод**

Как видно по графикам, функция аппроксимируется с помощью МНСП достаточно точно. Графики многочленов 5-ой и 6-ой степени совпадают с графиком исходной функции.