**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Лабораторная №4

**Интерполяция алгебраическими многочленами**

**Выполнил:**

Кендысь Алексей Максимович

студент 2 курса, 9 группы,

специальность

“прикладная математика”

**Преподавательница:**

Ассистентка кафедры вычислительной

математики ФПМИ,

Ю.Н. Горбачёва

Минск, 2022

**Содержание:**

Постановка задачи ------------------------------------------------------------------ 2

Краткие теоретические сведения ------------------------------------------------ 3

Листинг программы ---------------------------------------------------------------- 3-6

Результаты --------------------------------------------------------------------------- 6-15

Выводы ------------------------------------------------------------------------------- 15

**Постановка задачи**

На отрезке [a, b] заданы функции и . Постройте многочлены степени n = 5, 10, 15, 20, интерполирующие каждую из них:

а) на сетке равноотстоящих узлов;

б) на сетке чебышёвских узлов.

Постройте графики функции и интерполяционных многочленов для каждого n. Оцените погрешность интерполирования в узлах сетки . Сравните полученные результаты. Сделайте выводы о сходимости интерполяционного процесса по равноотстоящим и чебышёвским узлам. В содержание отчета должна быть включена следующая информация:

• Способ выбора узлов.

• Представление, использованное при построении интерполяционных многочленов.

• Графики функции и интерполяционных многочленов по равноотстоящим узлам. Графики функции и интерполяционных многочленов по чебышёвским узлам.

• Графики функции и интерполяционных многочленов по равноотстоящим узлам. Графики функции и интерполяционных многочленов по чебышёвским узлам.

• Оценка погрешности интерполирования функции , оформленная в виде таблицы.

• Оценка погрешности интерполирования функции , оформленная в виде таблицы 2.

• Листинг программы с комментариями.

Вариант 4.

Функции . Отрезок

**Краткие теоретические сведения**

*Выбор узлов:*

Для .

Равноотстоящие:

Чебышёвские:

*Представление алгебраического интерполяционного многочлена в форме Лагранжа:*

**Листинг программы**

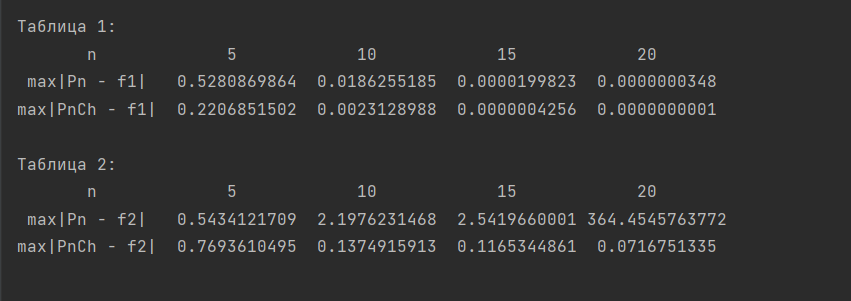
Файл Function.java:

import java.util.\*;  
  
public class Function {  
 private static final int *NUM\_OF\_POINT\_TYPES* = 2;  
 private static final int[] *N* = {5, 10, 15, 20};  
 private static final double *A* = -3.;  
 private static final double *B* = 3.;  
 private static final int *N\_XI* = 100;  
 private final double[] xi;  
 private final double[][] table1;  
 private final double[][] table2;  
 private double[] c1;  
 private double[] c2;  
 private double[] xk;  
  
 public Function() {  
 //Таблицы  
 table1 = new double[*NUM\_OF\_POINT\_TYPES*][*N*.length];  
 table2 = new double[*NUM\_OF\_POINT\_TYPES*][*N*.length];  
  
 //Вектор узлов сетки из условия  
 xi = new double[*N\_XI* + 1];  
 for(int i = 0; i < *N\_XI* + 1; i++) {  
 xi[i] = *A* + i \* (*B* - *A*) / *N\_XI*;  
 }  
 }  
  
 public void formPolynomials() {  
 //Формирование многочленов и таблиц  
 for (int i = 0; i < *N*.length; i++) {  
 xk = formXk(*N*[i]);  
 lagrangePol(xk);  
 table1[0][i] = getD1();  
 table2[0][i] = getD2();  
 System.*out*.print("\nМногочлены по равноотстоящим узлам для n = " + *N*[i] + ":\n");  
 outPs();  
 xk = formXkCh(*N*[i]);  
 lagrangePol(xk);  
 table1[1][i] = getD1();  
 table2[1][i] = getD2();  
 System.*out*.print("\nМногочлены по чебышёвским узлам для n = " + *N*[i] + ":\n");  
 outPs();  
 }  
 }  
  
 public void outTables() {  
 //Вывод таблиц  
 Formatter fmt = new Formatter();  
 fmt.format("\nТаблица 1:\n");  
 fmt.format("%8s %8d %8d %8d %8d \n", "n", *N*[0], *N*[1], *N*[2], *N*[3]);  
 fmt.format(" max|Pn - f1| %13.10f %13.10f %13.10f %13.10f\n", table1[0][0], table1[0][1], table1[0][2], table1[0][3]);  
 fmt.format("max|PnCh - f1| %13.10f %13.10f %13.10f %13.10f\n", table1[1][0], table1[1][1], table1[1][2], table1[1][3]);  
 fmt.format("\nТаблица 2:\n");  
 fmt.format("%8s %8d %8d %8d %8d \n", "n", *N*[0], *N*[1], *N*[2], *N*[3]);  
 fmt.format(" max|Pn - f2| %13.10f %13.10f %13.10f %13.10f\n", table2[0][0], table2[0][1], table2[0][2], table2[0][3]);  
 fmt.format("max|PnCh - f2| %13.10f %13.10f %13.10f %13.10f\n", table2[1][0], table2[1][1], table2[1][2], table2[1][3]);  
 System.*out*.println(fmt);  
 fmt.close();  
 }  
  
 private void lagrangePol(double[] xk) {  
 //Подсчёт коэффициентов в представлении Лагранжа  
 c1 = new double[xk.length];  
 c2 = new double[xk.length];  
 for(int i = 0; i < xk.length; i++) {  
 c1[i] = 1.;  
 for(int j = 0; j < xk.length; j++) {  
 if(j != i) {  
 c1[i] /= xk[i] - xk[j];  
 }  
 }  
 c2[i] = c1[i] \* f2(xk[i]);  
 c1[i] \*= f1(xk[i]);  
 }  
 }  
  
 private void outPs() {  
 //Вывод многочленов  
 Formatter fmt = new Formatter();  
 fmt.format("Для функции f1(x):\n");  
 fmtWithC(fmt, c1);  
 fmt.format("Для функции f2(x):\n");  
 fmtWithC(fmt, c2);  
 System.*out*.print(fmt);  
 fmt.close();  
 }  
  
 private void fmtWithC(Formatter fmt, double[] c) {  
 fmt.format("P(x) = ");  
 for(int i = 0; i < c.length; i++) {  
 fmt.format("(%.10f) ", c[i]);  
 for(int j = 0; j < c.length; j++) {  
 if(j != i) {  
 fmt.format("\* (x - (%.4f)) ", xk[j]);  
 }  
 }  
 if(i != c.length - 1) {  
 fmt.format("+ ");  
 }  
 }  
 fmt.format("\n");  
 }  
  
 private double f1(double x) {  
 //Функция f1(x)  
 return Math.*sin*(x) \* Math.*cos*(x);  
 }  
  
 private double f2(double x) {  
 //Функция f2(x)  
 return 1. / (1. + 12. \* Math.*pow*(x, 4.));  
 }  
  
 private double getD1() {  
 //Подсчёт max приращения для f1  
 double max = 0.;  
 for(int i = 0; i < *N\_XI* + 1; i++) {  
 if(Math.*abs*(P(xi[i], 1) - f1(xi[i])) > max) {  
 max = Math.*abs*(P(xi[i], 1) - f1(xi[i]));  
 }  
 }  
 return max;  
 }  
  
 private double getD2() {  
 //Подсчёт max приращения для f2  
 double max = 0.;  
 for(int i = 0; i < *N\_XI* + 1; i++) {  
 if(Math.*abs*(P(xi[i], 2) - f2(xi[i])) > max) {  
 max = Math.*abs*(P(xi[i], 2) - f2(xi[i]));  
 }  
 }  
 return max;  
 }  
  
 private double P(double x, int k) {  
 //Полученные многочлены, k - номер функции (1 или 2)  
 double res = 0.;  
 for(int i = 0; i < xk.length; i++) {  
 double m;  
 if(k == 1) {  
 m = c1[i];  
 } else {  
 m = c2[i];  
 }  
 for(int j = 0; j < xk.length; j++) {  
 if(j != i) {  
 m \*= x - xk[j];  
 }  
 }  
 res += m;  
 }  
 return res;  
 }  
  
 private double[] formXk(int n) {  
 //Вектор равноотстоящих узлов  
 double[] xk = new double[n + 1];  
 double h = (*B* - *A*) / n;  
 xk[0] = *A*;  
 for(int i = 1; i < n + 1; i++) {  
 xk[i] = xk[i - 1] + h;  
 }  
 return xk;  
 }  
  
 private double[] formXkCh(int n) {  
 //Вектор чебышёвских узлов  
 double[] xk = new double[n + 1];  
 for(int i = 0; i < n + 1; i++) {  
 xk[i] = ((*A* + *B*) / 2.) + ((*B* - *A*) / 2.) \* Math.*cos*((2 \* i + 1) \* Math.*PI* / (2 \* n + 2));  
 }  
 return xk;  
 }  
}

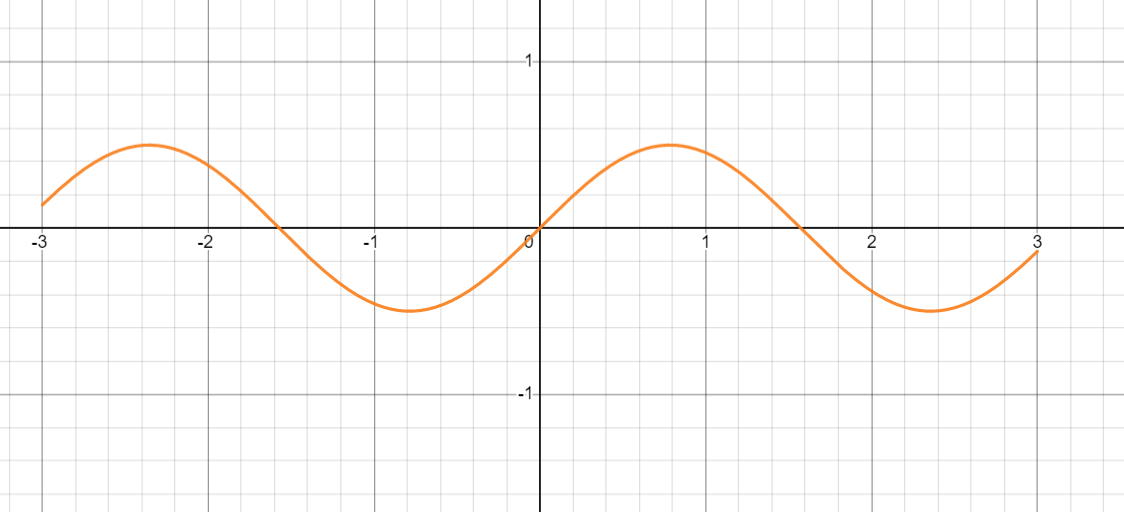
Файл Main.java:

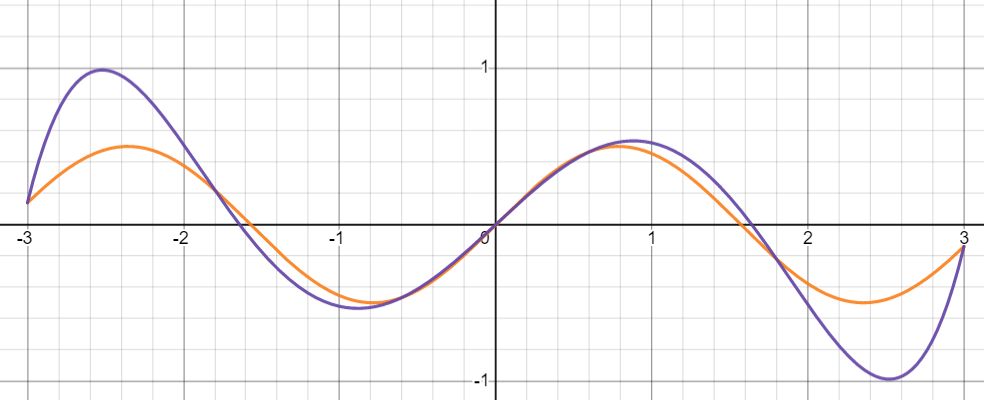
public class Main {  
  
 public static void main(String[] args) {  
 Function myF = new Function();  
 myF.formPolynomials();  
 myF.outTables();  
 }  
}

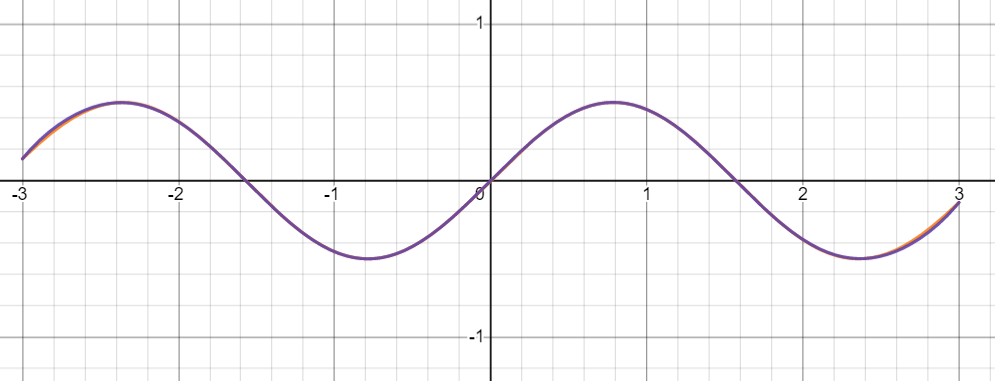
**Результаты**

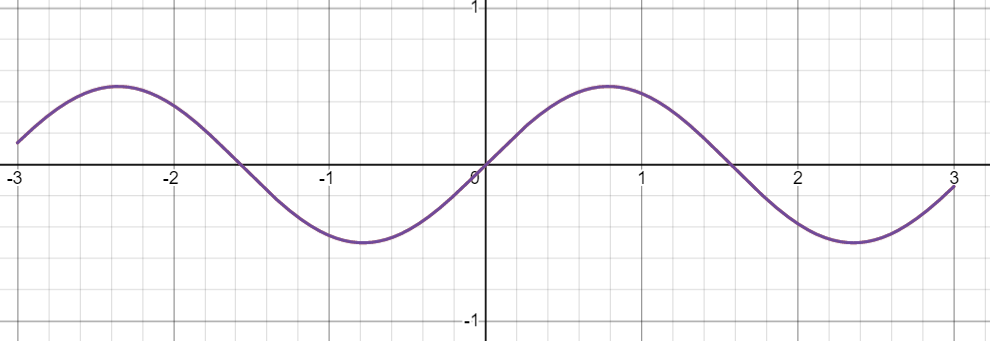
****

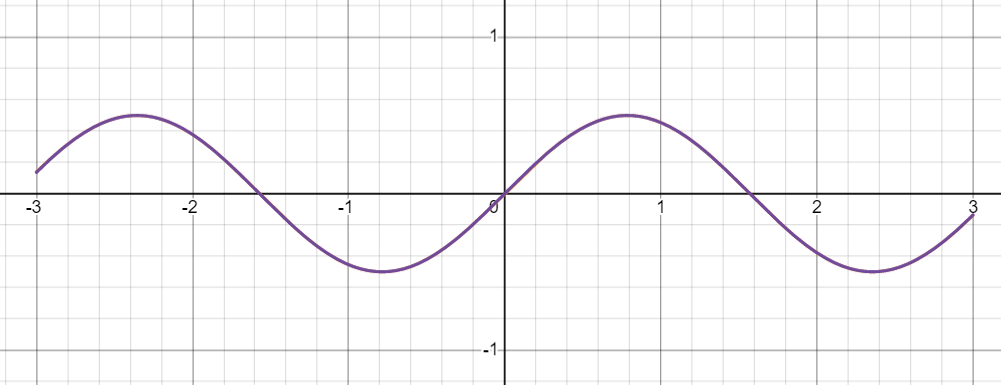
Графики:

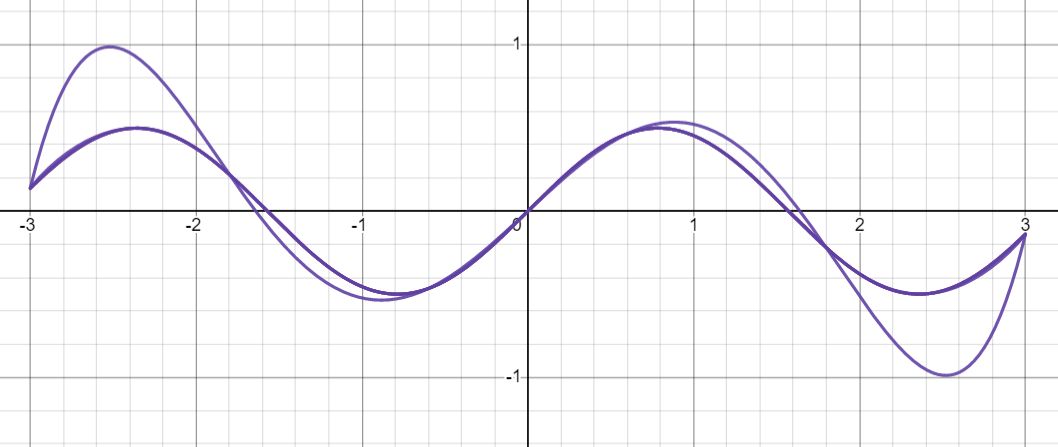


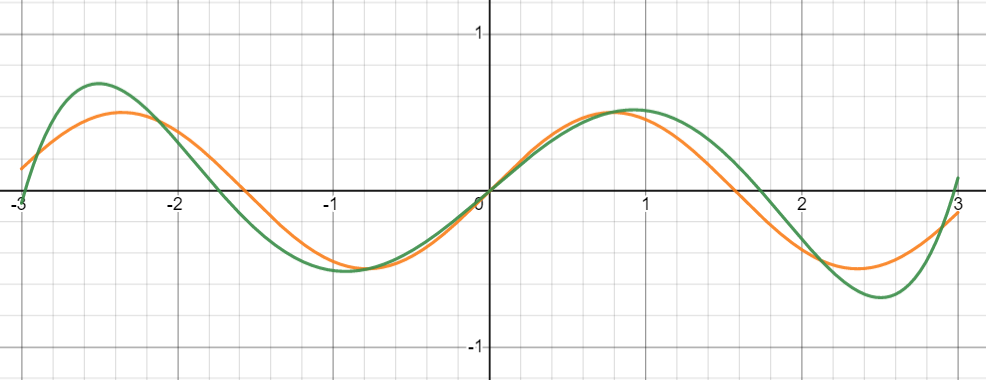


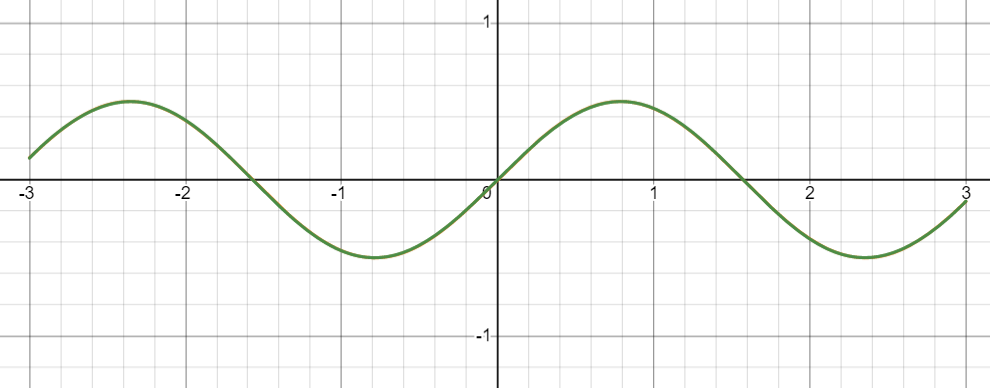


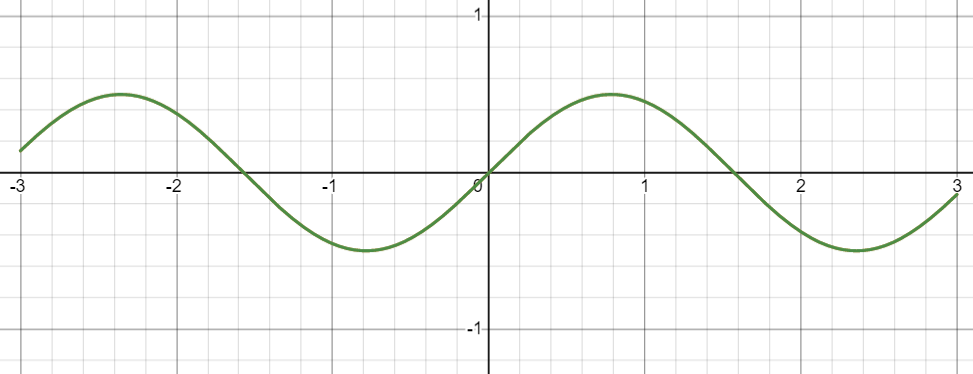


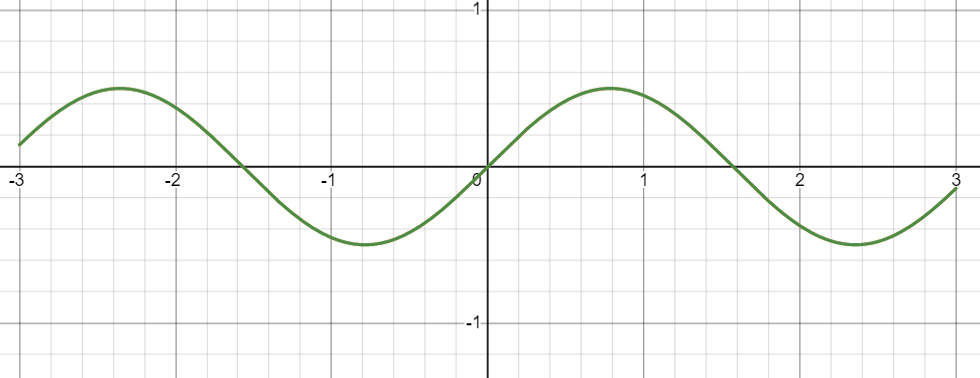


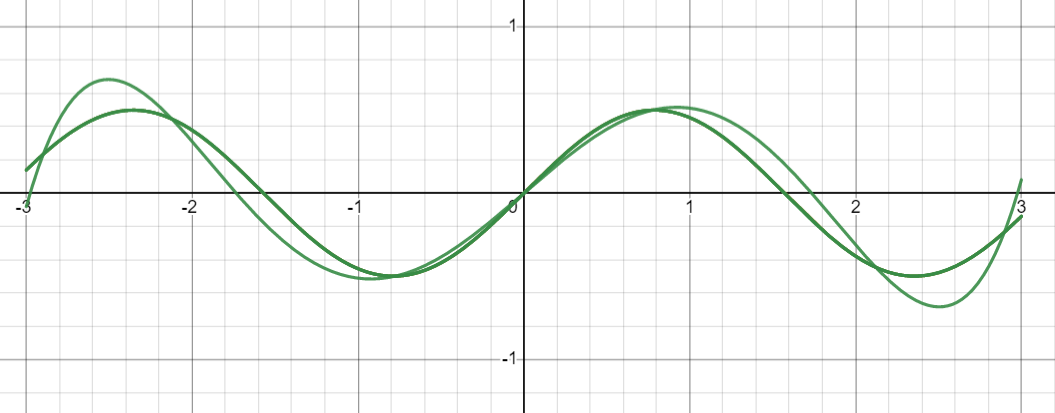


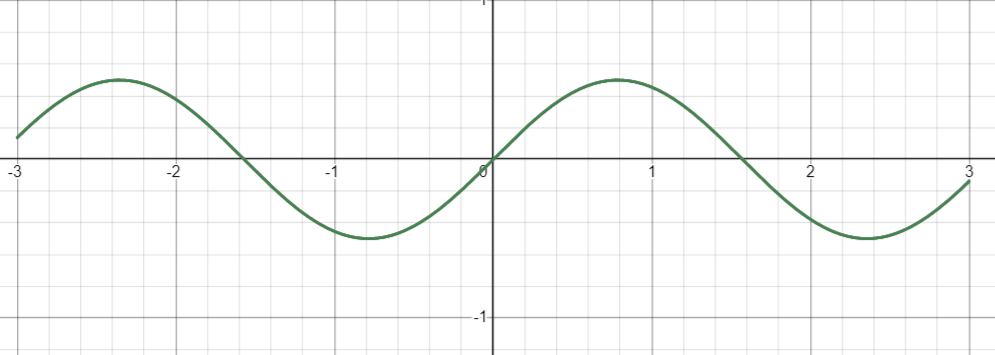


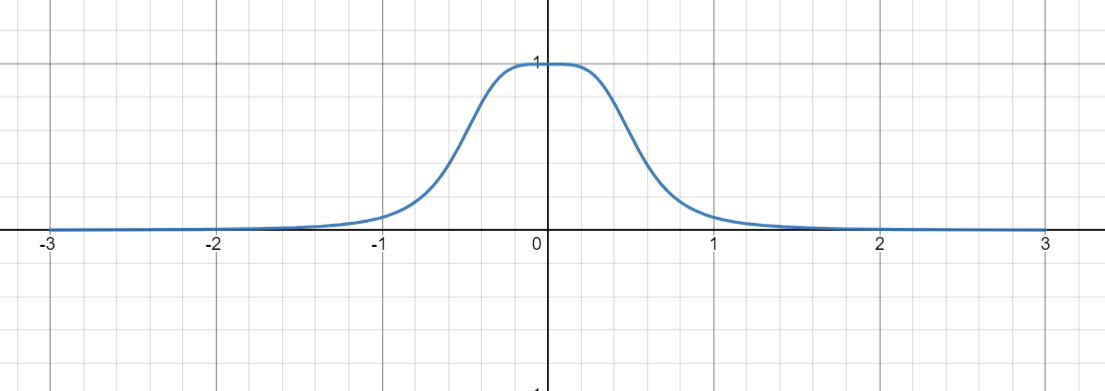


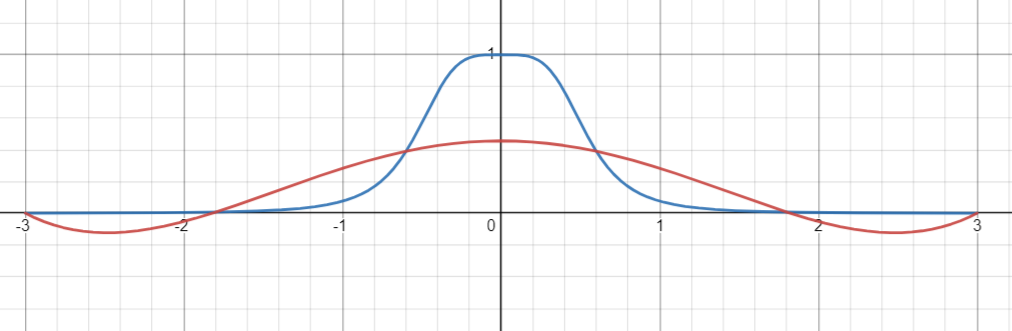


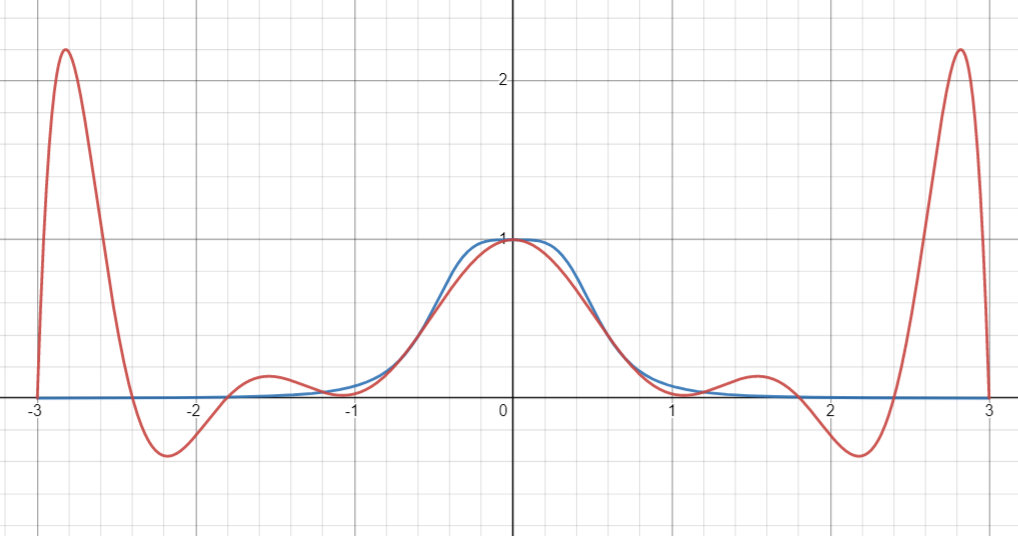


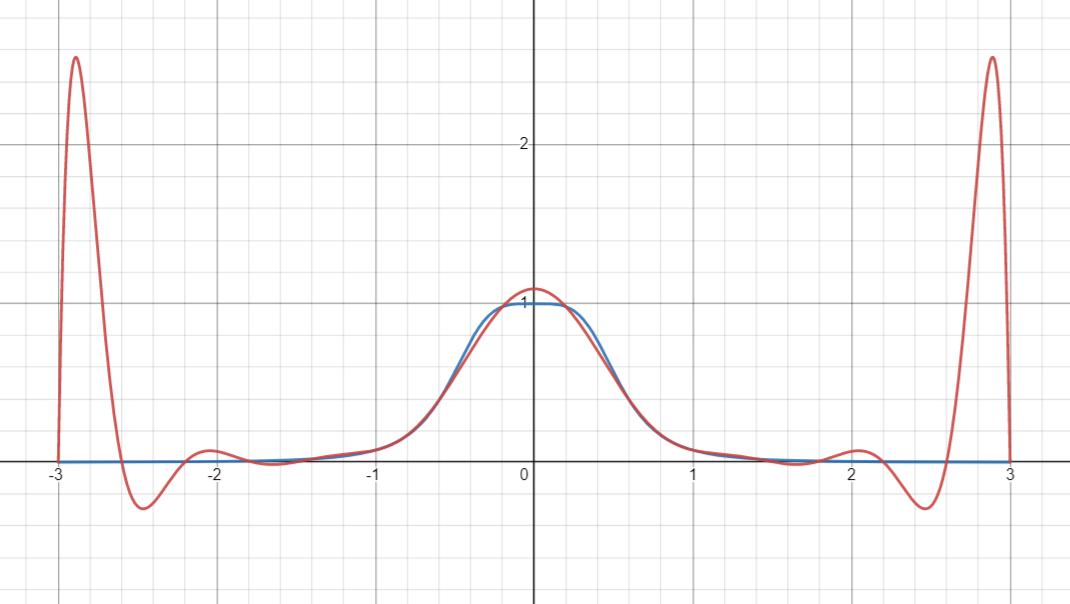


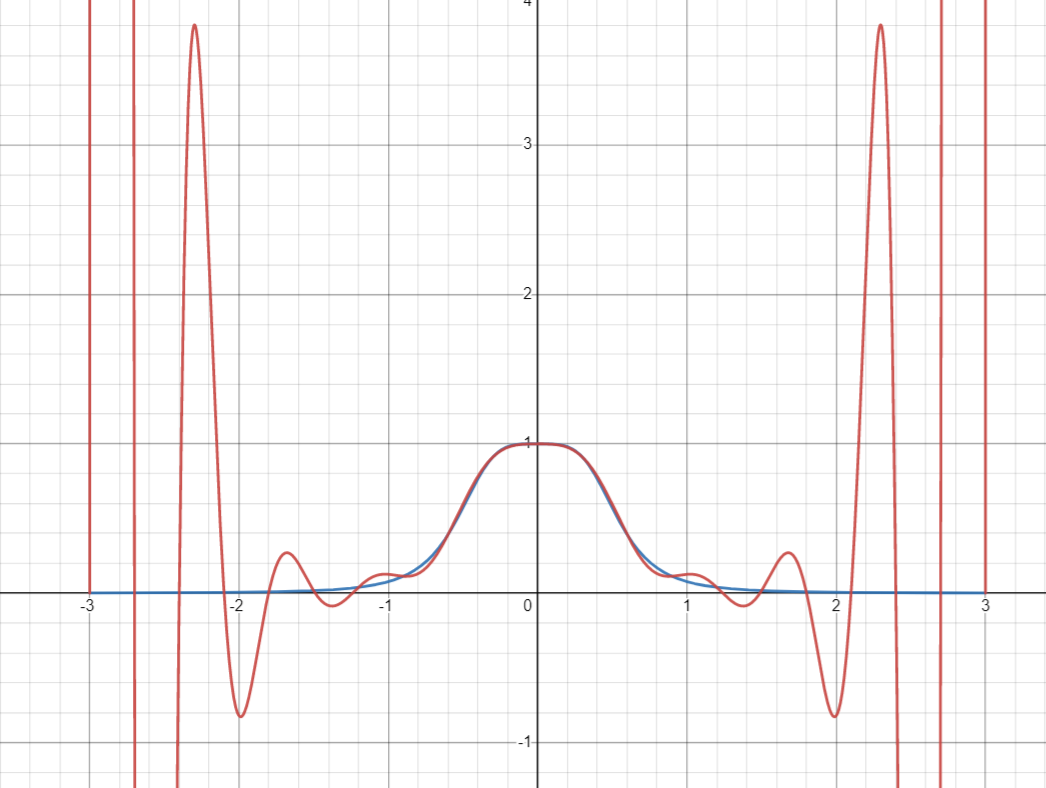


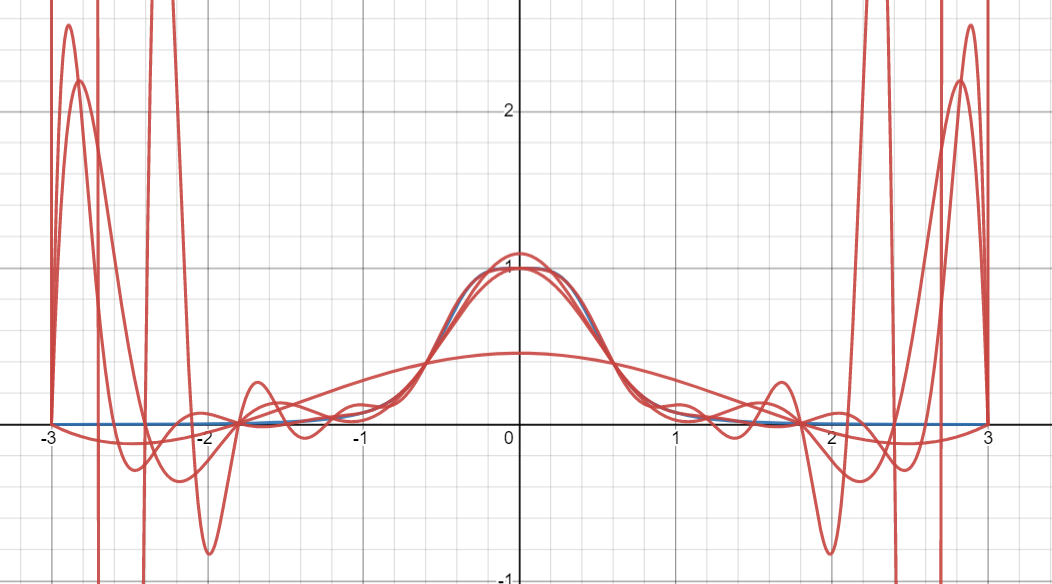


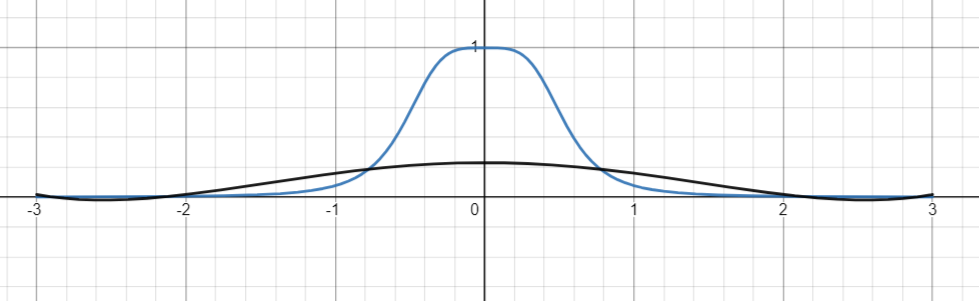


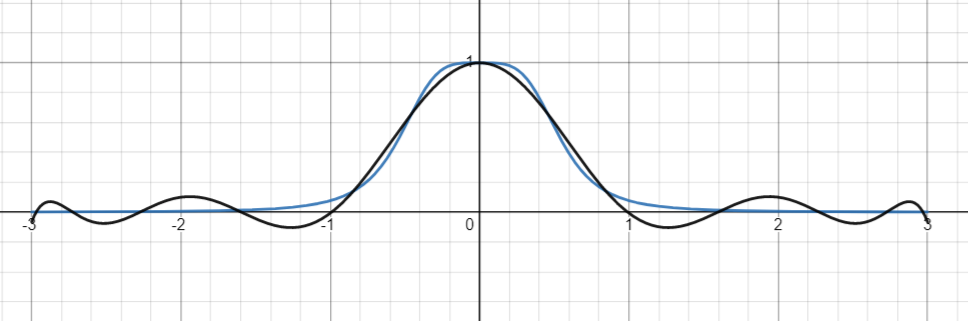


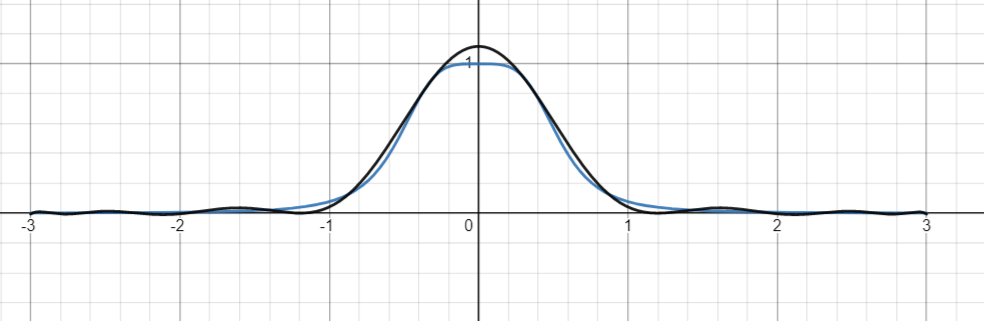


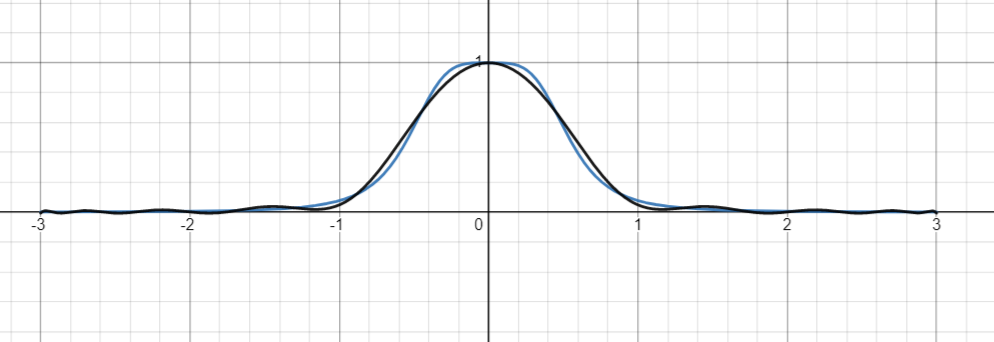


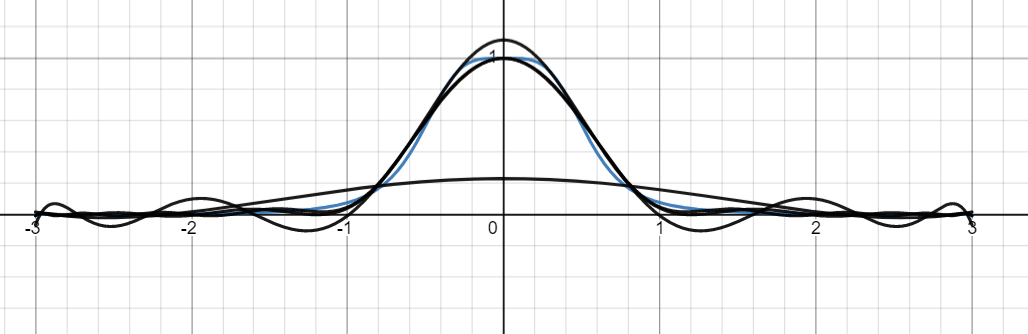


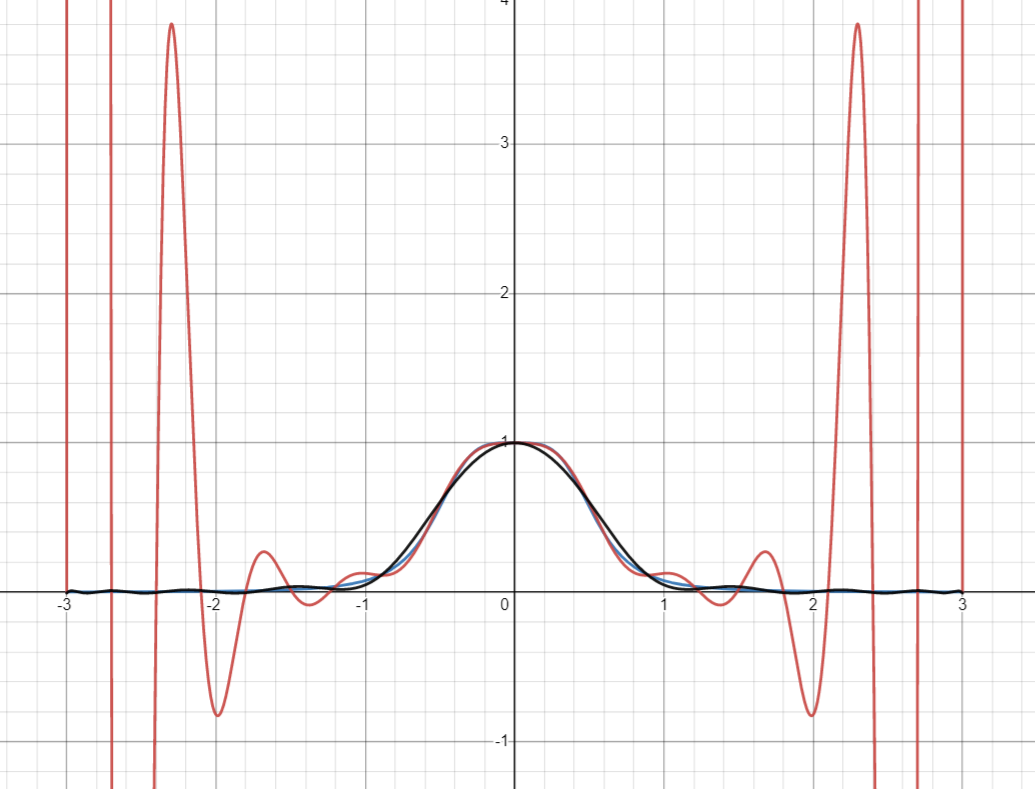












**Выводы**

1) Из таблиц видно, что по чебышёвским узлам интерполяционный процесс сходится быстрее, чем по равноотстоящим. К тому же интерполяционный процесс по чебышёвским узлам имеет больший шанс сходимости в целом.

2) Интерполяционный многочлен можно было также строить в форме Ньютона. Форма Ньютона чаще используется в случае, когда для одной функции нужно построить многочлены разных степеней (узлы просто добавляются). Форму Лагранжа же удобнее использовать в случае, когда узлы те же, но аппроксимируются разные функции.

3) Также интерполяционный многочлен можно построить по стандартному представлению многочлена (по определению), но для этого нужно решать систему. На практике этот метод не используется, т.к. при больших n система будет иметь большую обусловленность.