# Лабораторная работа 1

# Методы Вычислений

В данной лабораторной работе было 2 способа выбора узлов:

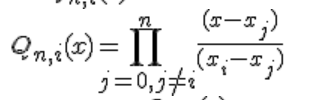
1. Равномерные узлы.

mult = a + i \* h;

1. Узлы Чебышева.

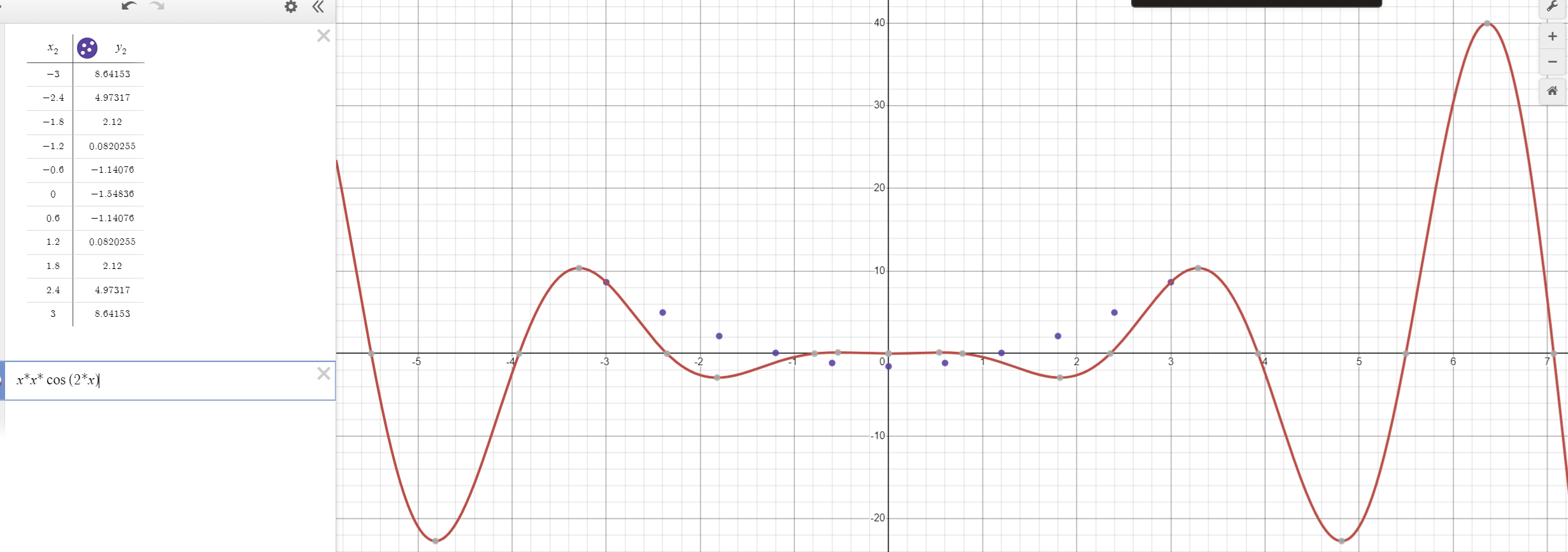
mult = (a + b) / 2.0 + (b - a) / 2.0 \* cos((2.0 \* i + 1.0) / (2.0\* (n + 1))\* acos(-1.0));

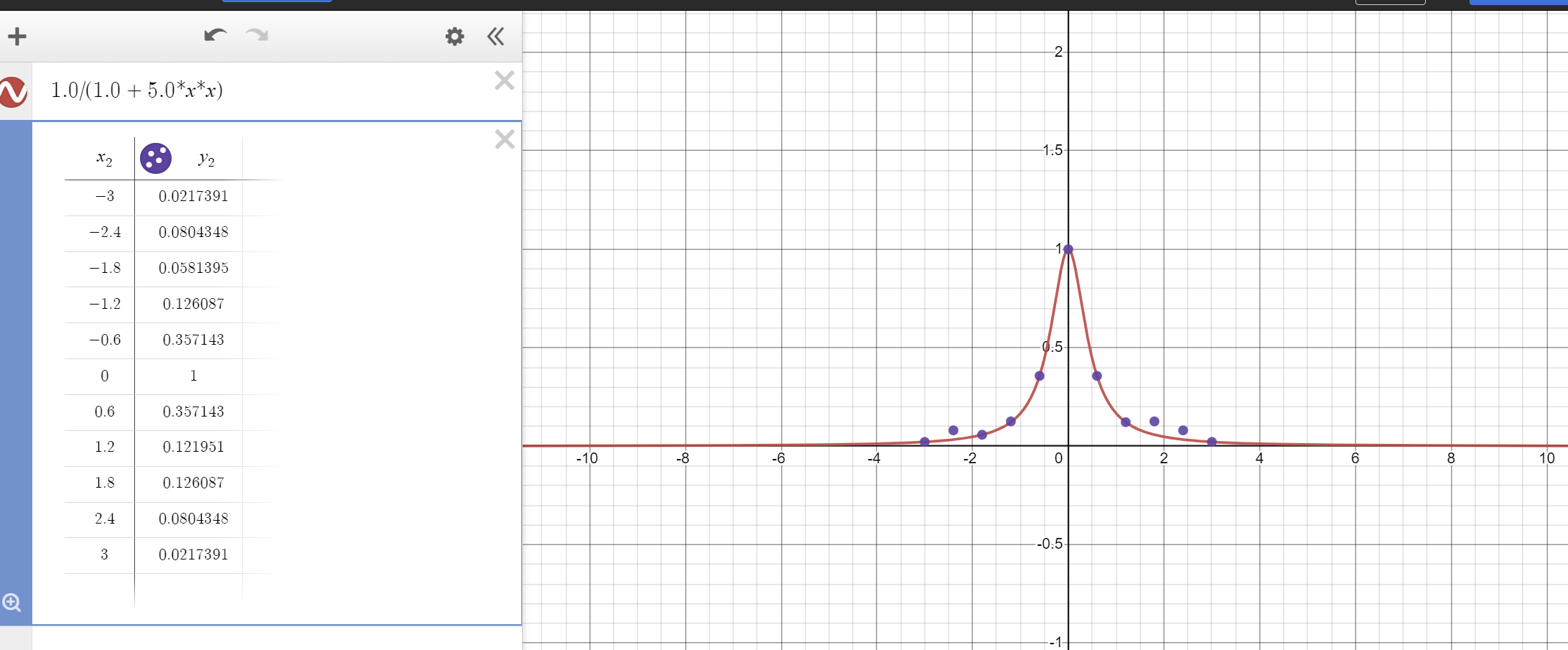
В данной программе для построения интерполяционных многочленов использовался многочлен Ньютона. Многочлен Ньютона представляет собой сумму членов, каждый из которых представляет произведение конечных разностей функции.



Графики функций:  
При n = 2;

Первой функции.

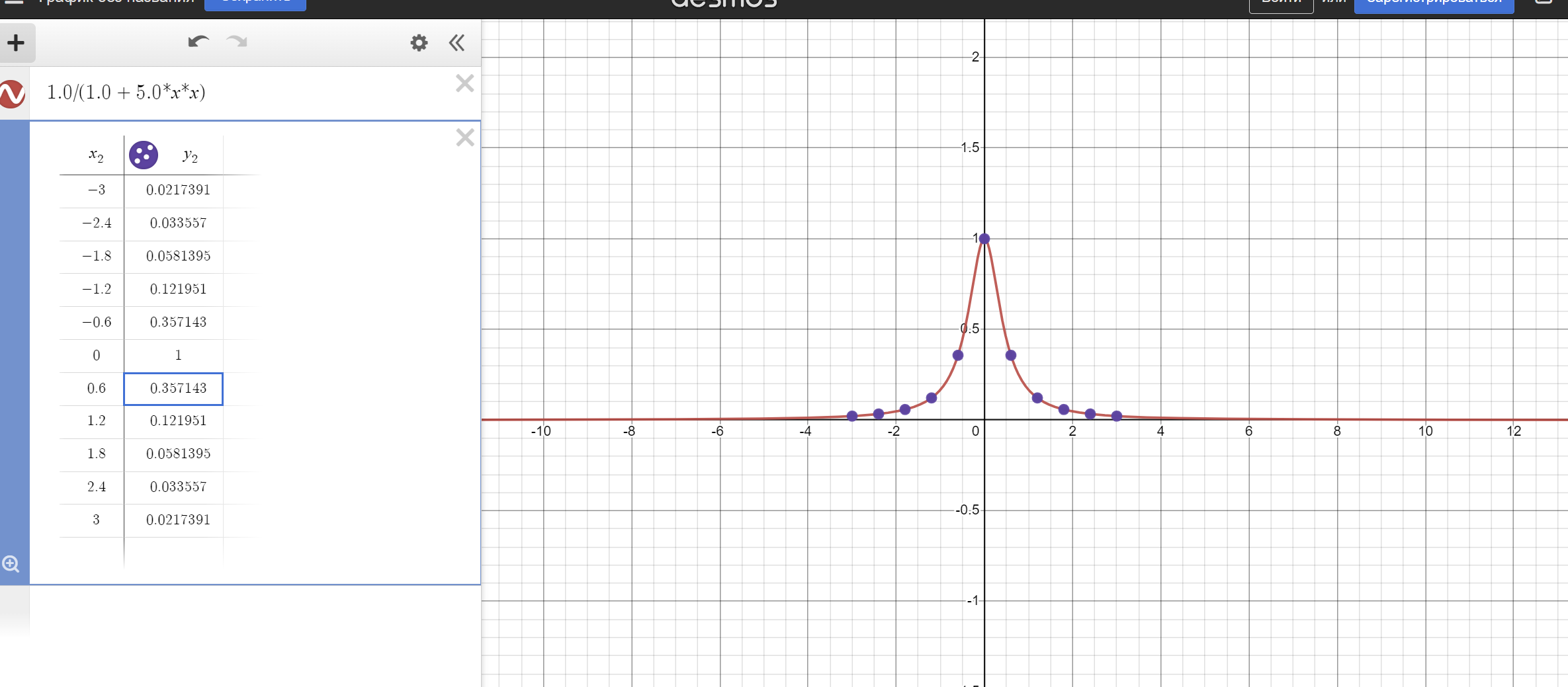


Второй функции:  


При n = 10;  
Первый график:



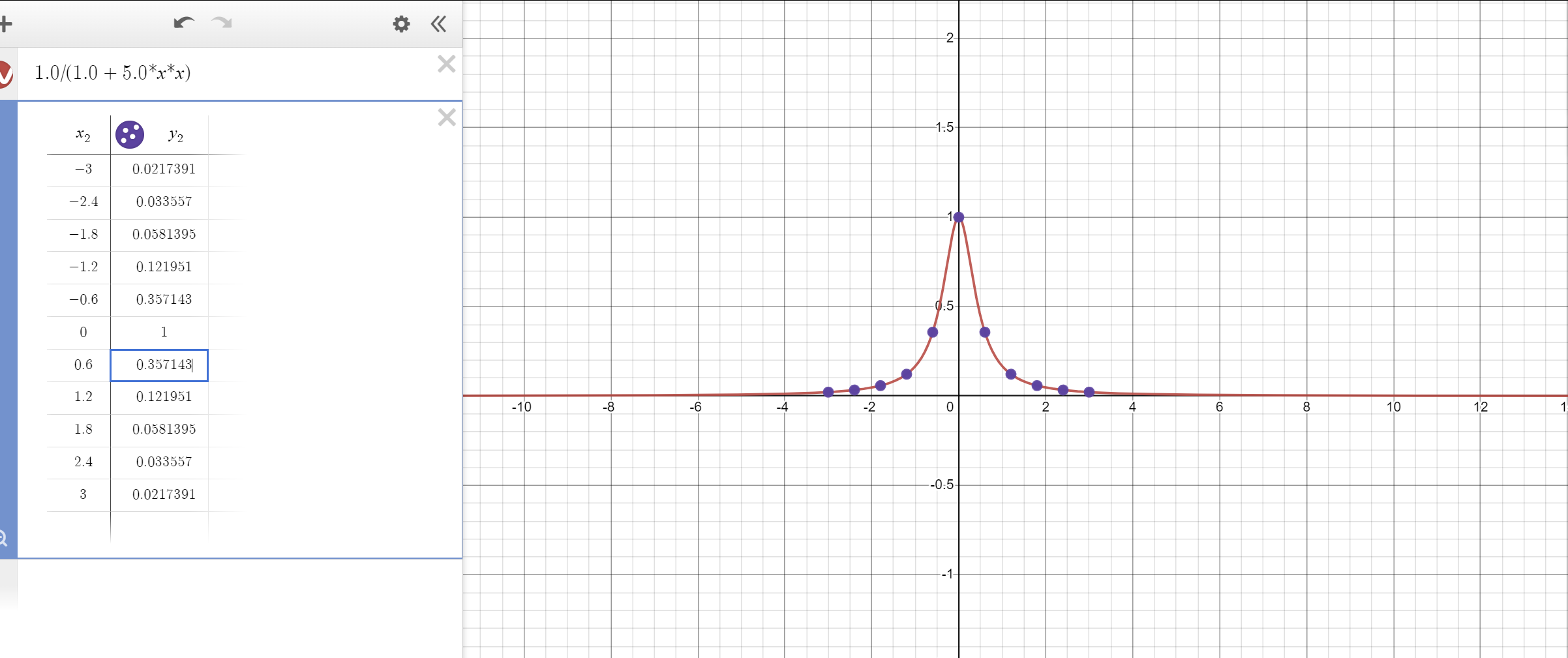
Второй:



При n = 30;



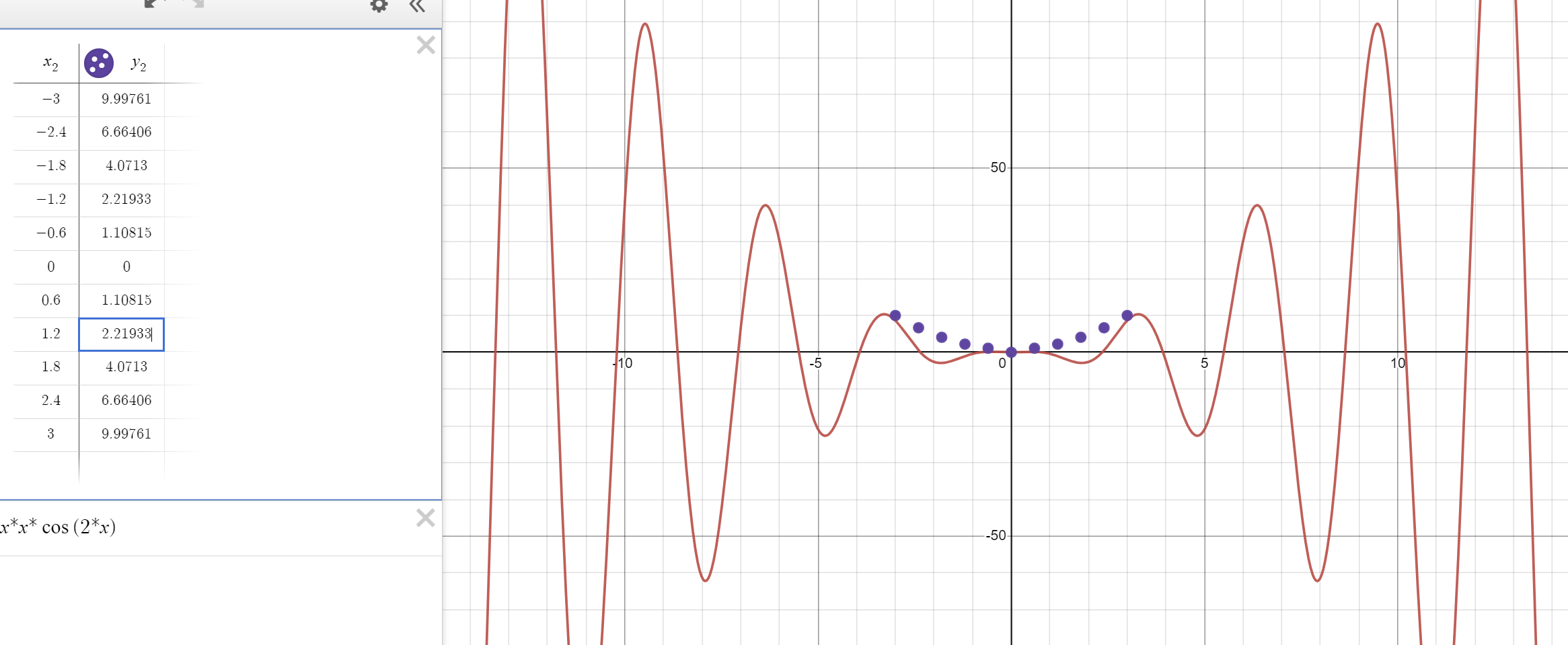
Второй:



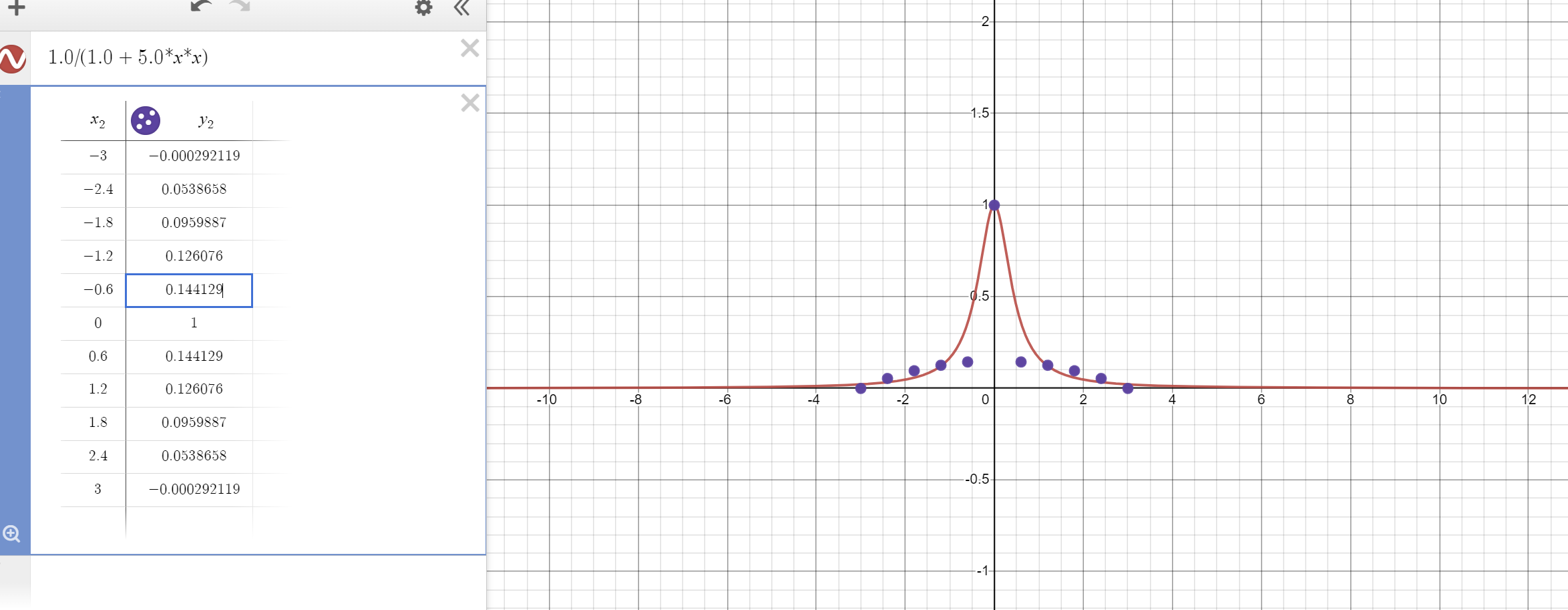
Перейдём к Узлам Чебышева:

При n = 3

Первая формула:

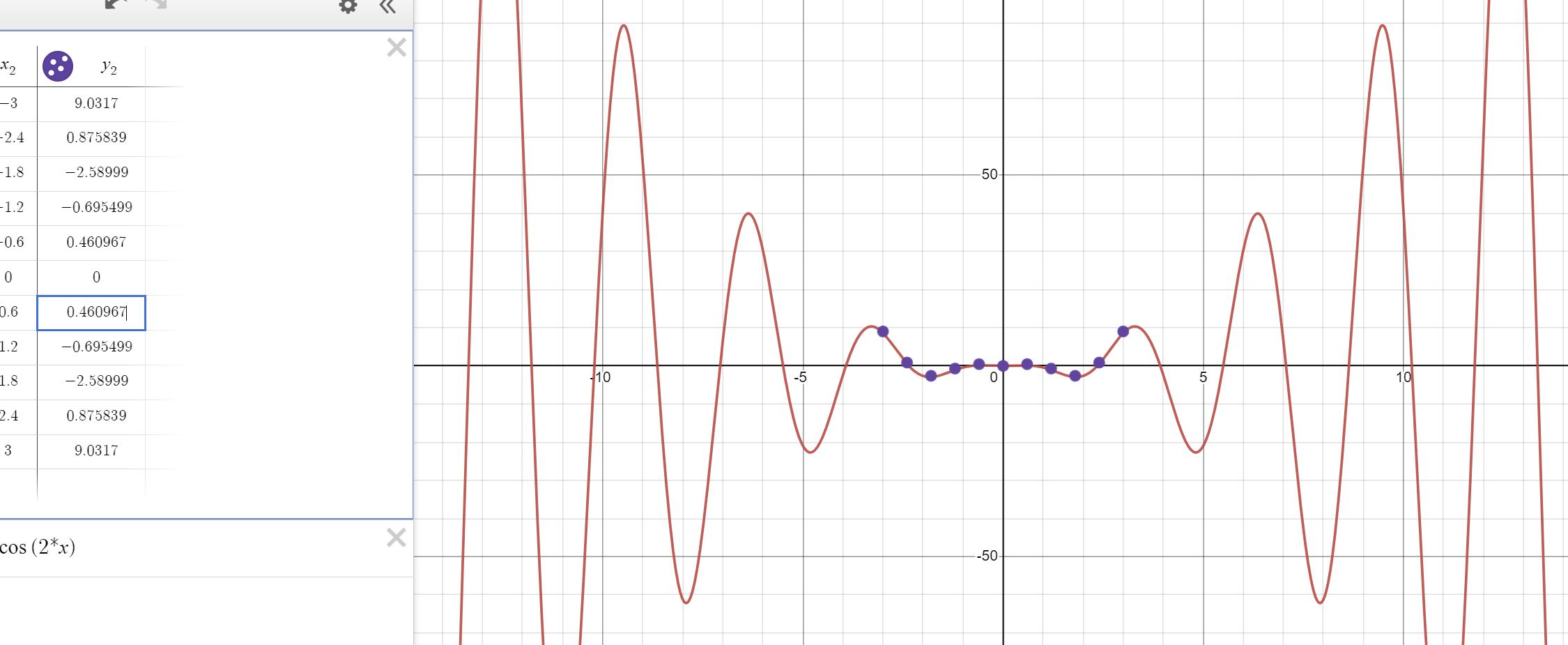


Вторая формула:

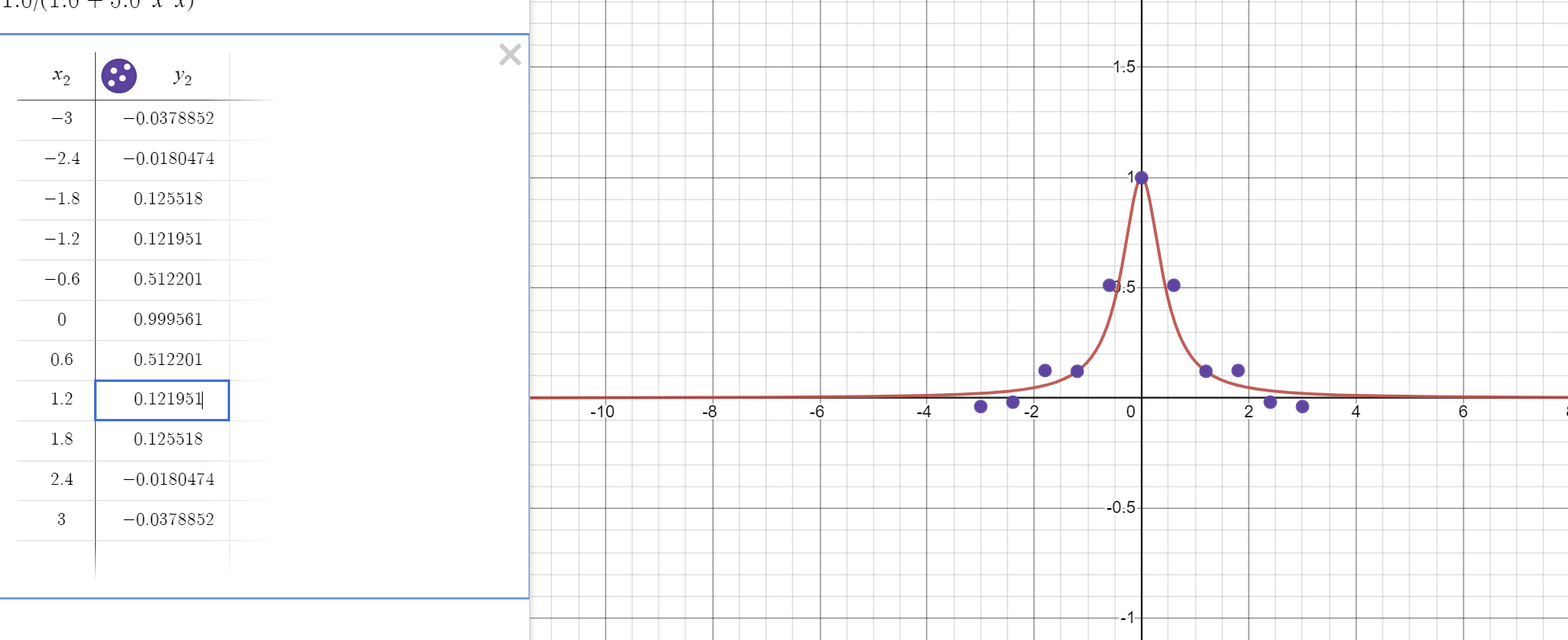


При n = 10

Первая формула:

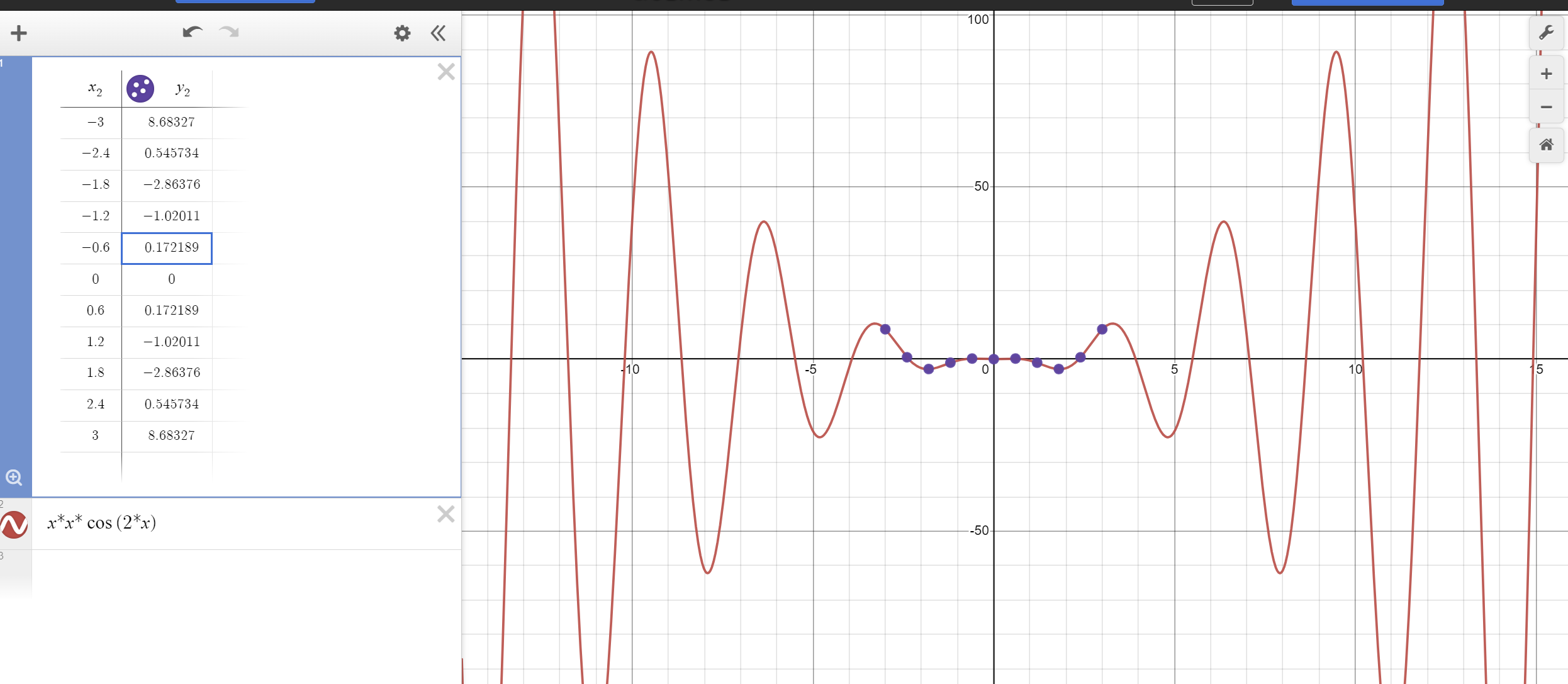


Вторая формула:

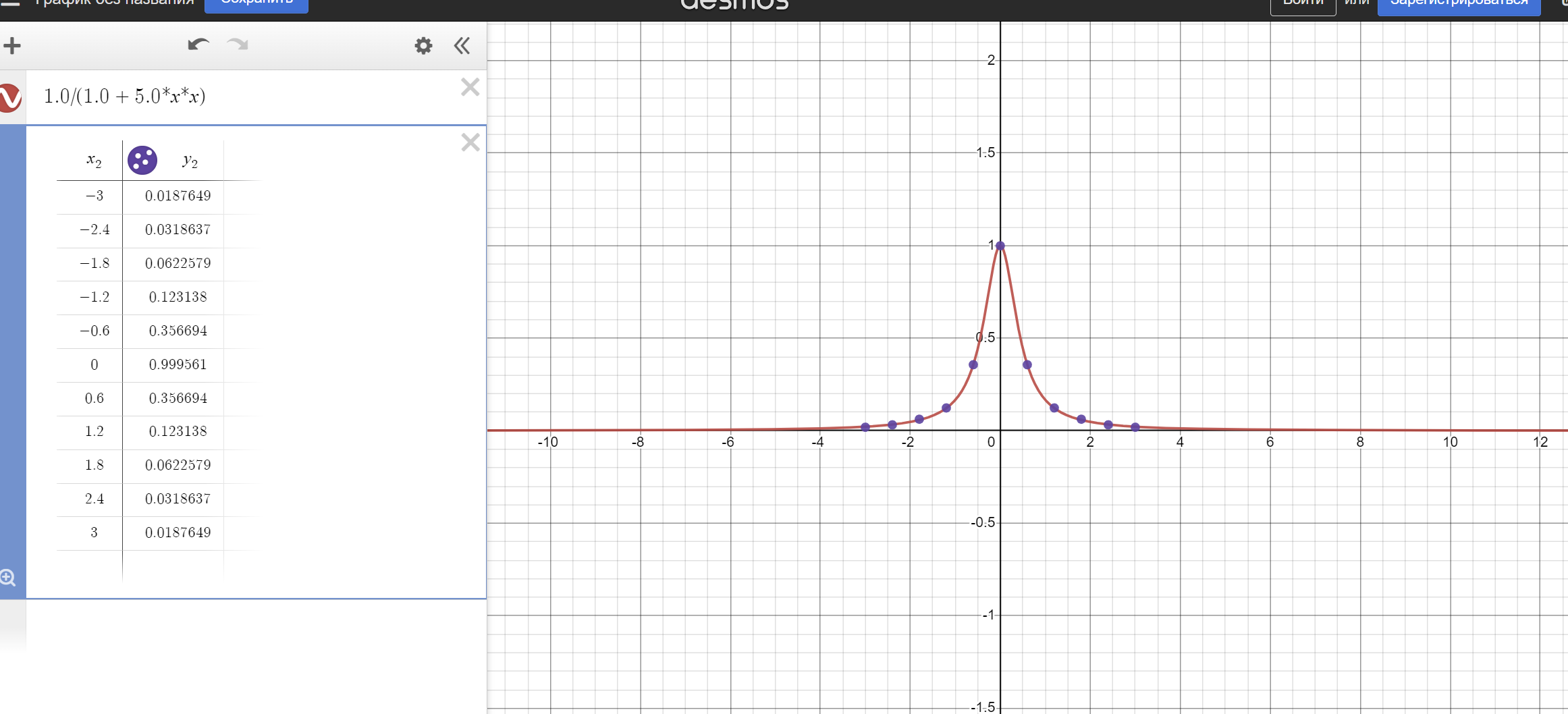


При n = 30

Первая формула:



Вторая формула:



Максимальные погрешности для первой и второй формул при равномерных узлах .

|  |  |  |
| --- | --- | --- |
| n | MaxP | MaxP |
| 3 | P+-=0.0001 | P+-=0.1 |
| 10 | P<0.000001 | P+-=0.0001 |
| 30 | P<0.000001 | P+-= 0.000003 |

Максимальные погрешности для первой и второй формул при узлах Чебышева .

|  |  |  |
| --- | --- | --- |
| n | MaxP | MaxP |
| 3 | P+-=0.03 | P+-=0.03 |
| 10 | P+-=0.02 | P+-=0.03 |
| 30 | P+-=0.01 | P+-=0.01 |

Обычно узлы Чебышева приводят к более быстрой сходимости интерполяционного процесса по сравнению с равномерно распределенными узлами. Это связано с тем, что узлы Чебышева минимизируют максимальное значение абсолютной величины многочлена Чебышева, который может использоваться для оценки ошибки интерполяции.

Но в моём случае получилось как-то не так(

Листенинг программы:

Программа проводит интерполяцию для функций f1 и f2 с использованием равноотстоящих и узлов Чебышёва, а затем выводит результаты.

Определение функций:

double f1(double x) {

return x \* x \* cos(2 \* x);

}

double f2(double x) {

return 1.0 / (1.0 + 5.0 \* x \* x);

}

Вычисление конечных разностей:

double calculateFiniteDifference(double x0, double x1, double f\_x0, double f\_x1) {

return (f\_x1 - f\_x0) / (x1 - x0);

}

Заполнение вектора конечных разностей:

void vecFiller(vector<vector<double>>& vec, int n, double a, double h, double (\*f)(double)) {

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

{

double mult = a + i \* h;

vec[i][0] = mult;

vec[i][1] = f(mult);

}

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

{

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

{

double finite\_difference = calculateFiniteDifference(vec[j][0], vec[i + j][0], vec[j][i], vec[j + 1][i]);

vec[j][i + 1] = finite\_difference;

}

}

}

Такая же функция есть для узлов Чебышева .

Интерполяционные полиномы для равноотстоящих узлов:

double calculateInterpolationPolynomial(double x, double x0, double b, double h, double f\_x0, int n, double (\*f)(double), vector<vector<double>> vec) {

double result = f(x0);

double term = 1.0;

double mult = 0;

double test = 0;

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

mult = x - (((x0 + b) / 2.0 + (b - x0) / 2.0 \* cos(acos(-1.0) \* (2.0 \* (i - 1) + 1.0) / (2.0 \* (n + 1)))));

term \*= mult;

test = vec[0][i + 1];

result += test \* term;

}

return result;

}

Интерполяционные полиномы для узлов Чебышёва:

double calculateInterpolationPolynomial2(double x, double x0, double b, double h, double f\_x0, int n, double (\*f)(double), vector<vector<double>> vec){

double result = f(x0);

double term = 1.0;

double mult = 0;

double test = 0;

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

mult = x - (((x0 + b) / 2.0 + (b - x0) / 2.0 \* cos(acos(-1.0) \* (2.0 \* (i - 1) + 1.0) / (2.0 \* (n + 1)))));

term \*= mult;

test = vec[0][i + 1];

result += test \* term;

}

return result;

}

Основная функция main:

int main() {

// Инициализация параметров

// ...

// Заполнение и вычисление для f1

// ...

// Заполнение и вычисление для f2

// ...

// Вывод результатов

// ...

return 0;

}