Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное   
учреждение высшего образования

Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по лабораторной работе**

**«Вычисление математических функций»**

**Выполнил**:

студент группы 3823Б1ПМ1-1

Золкин И.А.

**Проверил**:

преподаватель каф. ВВСП,

Волокитин В.Д.

Нижний Новгород

2024

**Содержание**

[Постановка задачи 3](#_Toc26962562)

[Метод решения 4](#_Toc26962563)

[Руководство пользователя 5](#_Toc26962564)

[Описание программной реализации 6](#_Toc26962565)

[Подтверждение корректности 9](#_Toc26962566)

[Результаты экспериментов 11](#_Toc26962567)

[Заключение 13](#_Toc26962568)

[Список литературы 19](#_Toc1)

[Приложение 20](#_Toc26962569)

# Постановка задачи

Программа, выполненная на языке программирования C, должна корректно находить значение математических функций через ряд Маклорена (в окрестности нуля) по данным формулам:

Необходимо реализовать 3 варианта данного алгоритма:

* Методом прямого суммирования,
* Методом обратного суммирования,
* Методом прямого суммирования пар.

Нужно сравнить данные методы между собой и выявить вариант, возвращающий результат с наименьшей ошибкой.

# Метод решения

Метод прямой суммы:

Алгоритм последовательно суммирует члены ряда от нулевого до заданного включительно. Заводится переменная a, в которую при запуске записывается нулевой член ряда. Запускается цикл, в котором на каждой итерации вычисляется множитель по заданной функции, этот множитель перемножается на a, формируя i-й член ряда. Результат формируется последовательной суммой вычисленных значений.

Метод обратной суммы:

Алгоритм последовательно вычисляет члены ряда способом, описанным в методе прямой суммы. Вычисленные значения записываются в массив. Идем по массиву с конца и суммируем записанные в i-й ячейке значения.

Метод прямого суммирования пар:

Запускается цикл с 0 до n+1, с шагом 2 и заводится две переменные для хранения двух соседних членов ряда. Если это первая итерация цикла, тогда в первую переменную записывается нулевой член, а во вторую первый. Иначе в первую переменную записывается i-й член ряда, во вторую - i+1-й. Т.к. Две переменные хранят два последовательных члена ряда, и на следующей итерации эти переменные уже хранят два следующих члена ряда, то значения переменных можно вычислять перемножением вычисленного множителя и переменной предыдущего члена. После вычисления двух последовательных членов, они складываются и прибавляются к переменной результата. При вычислении второй переменной происходит проверка, если это n+1 член ряда, тогда в переменную записывается 0.

# Руководство пользователя

Все три метода представлены в одном файле. Взаимодействие с пользователем происходит через консоль. На вход принимается число x, записанное современной арабской записью (цифрами) слева направо, от которого будут вычислены 4 функции: sin, cos, exp, log(1+x). Допустимы отрицательные, положительные, целые и нецелые значения, записанные в десятичной записи через запятую. Вывод программы будет содержать результаты вычислений математических функций встроенным алгоритмом и предложенными алгоритмами, а также ошибку вычисления для нескольких x.

# Описание программной реализации

Каталог содержит несколько файлов, среди которых кода “\*.cpp” и файл проекта “.sln”.

Программа начинает работу с функции main(). Внутри реализовано считывание числа x, вызов функций вычисления математических функций от числа x и вывод результатов в консоль.

Всего в программе 8 функций:

* Функции вычисления множителей для получения последующих членов ряда: float sin\_p() - для синуса, float cos\_p() - для косинуса, float exp\_p() - для экспоненты, float log\_p() - для логарифма.
* Функции вычисления математических функций через ряд Маклорена: float sum\_series\_forward() - метод прямой суммы, float sum\_series\_backward() - метод обратной суммы, float sum\_series\_pair\_forward() - метод прямого суммирования пар.
* Функция, вызываемая при запуске программы: int main().

Функция float sin\_p(float x, int i) запускается из функций вычисления математических функций. В качестве аргументов передается значение переменной x, относительно которой идут вычисления и номер члена ряда, для которого вычисляется множитель. Функция находит множитель по формуле .

Функция float cos\_p(float x, int i) запускается из функций вычисления математических функций. В качестве аргументов передается значение переменной x, относительно которой идут вычисления и номер члена ряда, для которого вычисляется множитель. Функция находит множитель по формуле .

Функция float log\_p(float x, int i) запускается из функций вычисления математических функций. В качестве аргументов передается значение переменной x, относительно которой идут вычисления и номер члена ряда, для которого вычисляется множитель. Функция находит множитель по формуле .

Функция float exp\_p(float x, int i) запускается из функций вычисления математических функций. В качестве аргументов передается значение переменной x, относительно которой идут вычисления и номер члена ряда, для которого вычисляется множитель. Функция находит множитель по формуле .

Функция float sum\_series\_forward(float a, float x, float(\*next)(float, int), int n) запускается из main(), принимает на вход 4 аргумента:

* float a – нулевой член ряда.
* float x – значение от которого вычисляется функция.
* float(\*next)(float,int) – функция вычисления множителей для получения последующих членов ряда.
* int n – номер члена до которого идет вычисление ряда Маклорена (включительно).

Заводится переменная для хранения результата float , которой присваивается значение нулевого члена a. Дальше запускается цикл for{…} от 1 до , где на каждой итерации вычисляется текущий () член ряда и записывается в переменную a по формуле . Потом a прибавляется к переменной res. Функция возвращает float .

Функция float sum\_series\_backward(float a, float x, float(\*next)(float, int), int n) запускается из main(), принимает на вход 4 аргумента:

* float a – нулевой член ряда.
* float x – значение от которого вычисляется функция.
* float(\*next)(float,int) – функция вычисления множителей для получения последующих членов ряда.
* int n – номер члена до которого идет вычисление ряда Маклорена (включительно).

Заводится переменная для хранения результата float , которой присваивается значение нулевого члена . Создается массив для хранения членов ряда, в начало массива записывается нулевой член. Дальше запускается цикл for{…} от 1 до , где на каждой итерации вычисляется текущий () член ряда и записывается в переменную a по формуле , вычисленная переменная a записывается в . После вычисления всех требуемых членов ряда, запускаем цикл for{…} по массиву с конца и последовательно суммируем значения из массива в переменную . Функция возвращает float .

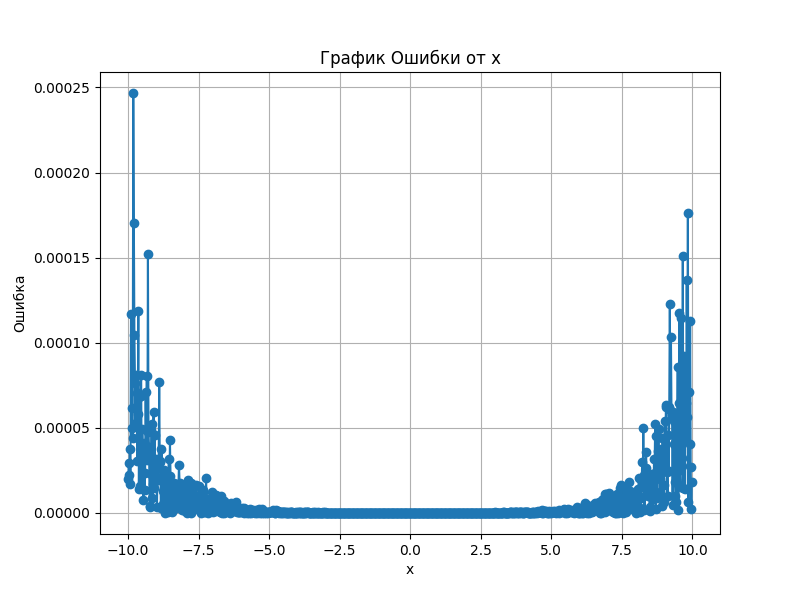
Функция float sum\_series\_pair\_forward(float a, float x, float(\*next)(float, int), int n) запускается из main(), принимает на вход 4 аргумента:

* float a – нулевой член ряда.
* float x – значение от которого вычисляется функция.
* float(\*next)(float,int) – функция вычисления множителей для получения последующих членов ряда.
* int n – номер члена до которого идет вычисление ряда Маклорена (включительно).

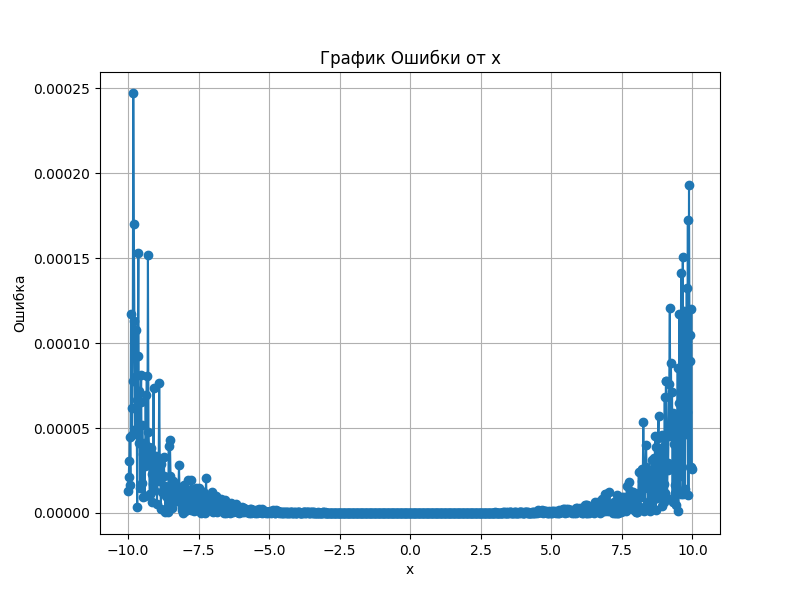
Заводится переменная для хранения результата float равная 0. Также заводятся две переменные float для хранения соседних членов ряда и переменная float для хранения промежуточной суммы. Дальше запускается цикл for{…} от 0 до , где на каждой итерации вычисляется текущий () член ряда и записывается в переменную по формуле , если это не нулевая итерация цикла, иначе в записывается значение нулевого члена a. Проверяем, нужен ли нам следующий член ряда, если да – в записываем значение члена по формуле , иначе – считаем . В переменную записываем и прибавляем к res переменную . Функция возвращает float res.

# Подтверждение корректности

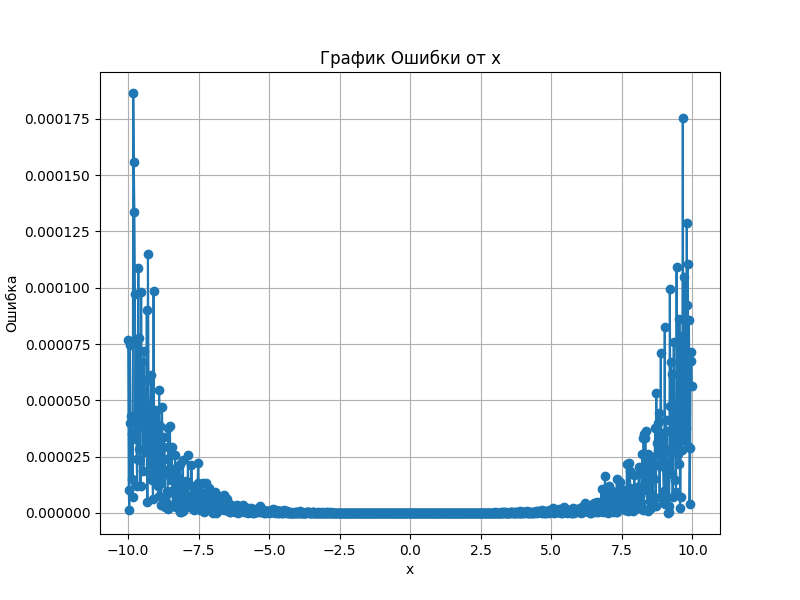
Для подтверждения корректности работы программ, использовалось сравнение вышеописанных функций с встроенными. Проверка проводилась на отрезке [ -10 ; 10 ] с шагом 0.02. Для каждого теста выводилась разница между результатами реализованных в программе функций и результатами встроенных функций. Из графиков ошибок видно sin(x) - [Рисунок 1, Рисунок 2, Рисунок 3], log(x+1) – [Рисунок 4, Рисунок 5, Рисунок 6], cos(x) – [Рисунок 7, Рисунок 8, Рисунок 9], exp(x) – [Рисунок 10, Рисунок 11, Рисунок 12], что функции дают верный результат вблизи нуля. Это связано с тем, что ряд Тэйлора (Маклорена) аппроксимирует с высокой точностью функцию в окрестности нуля.



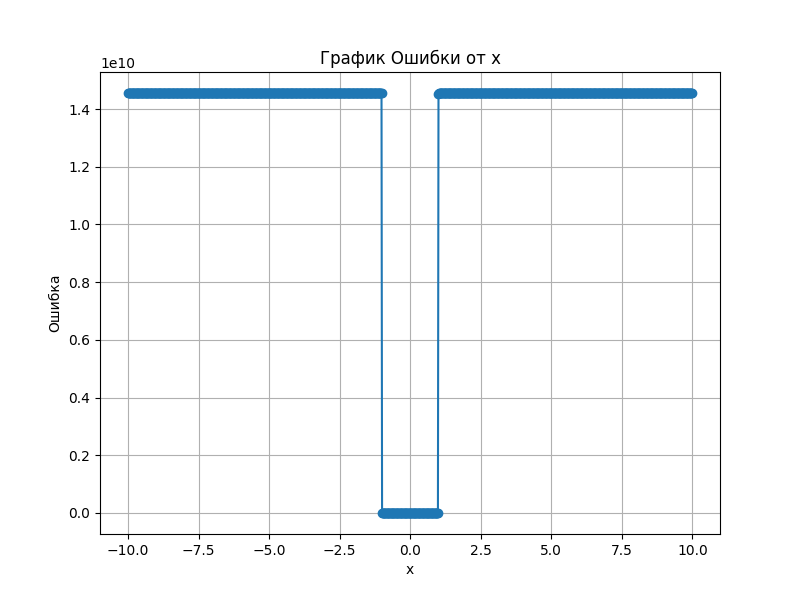
Рисунок



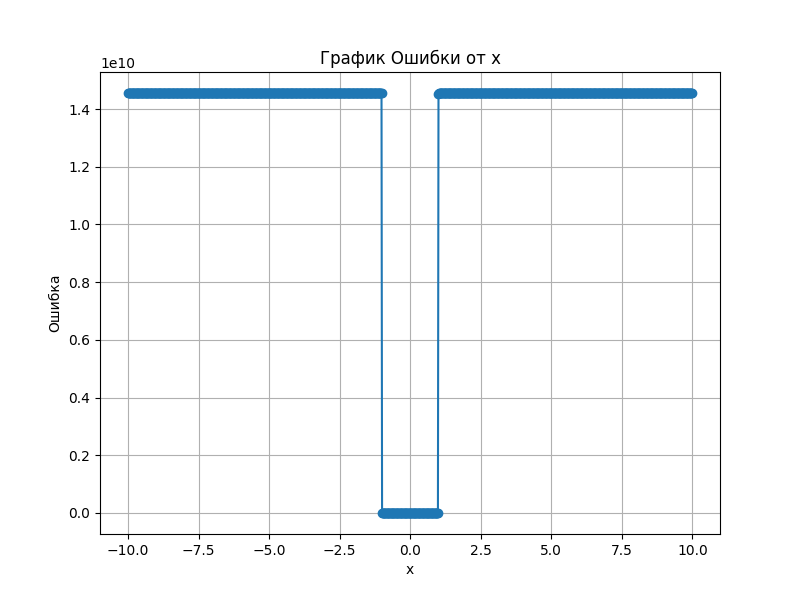
Рисунок



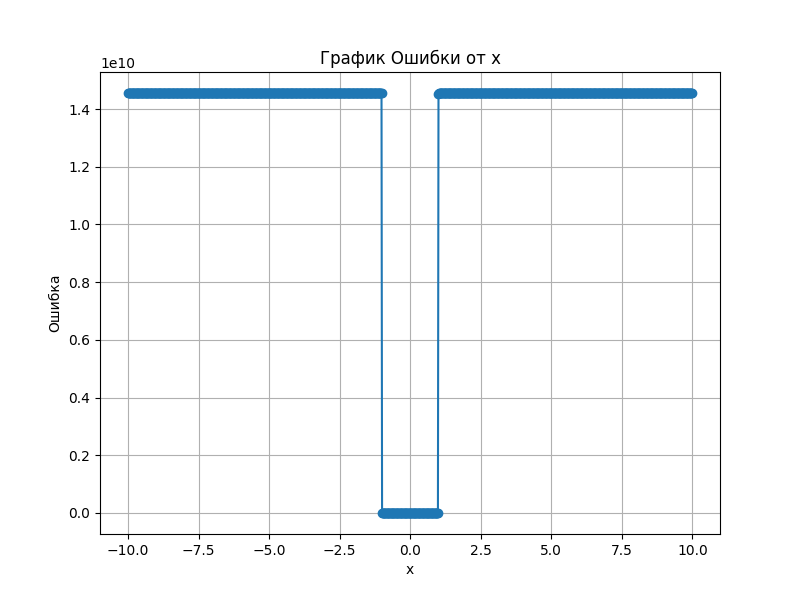
Рисунок



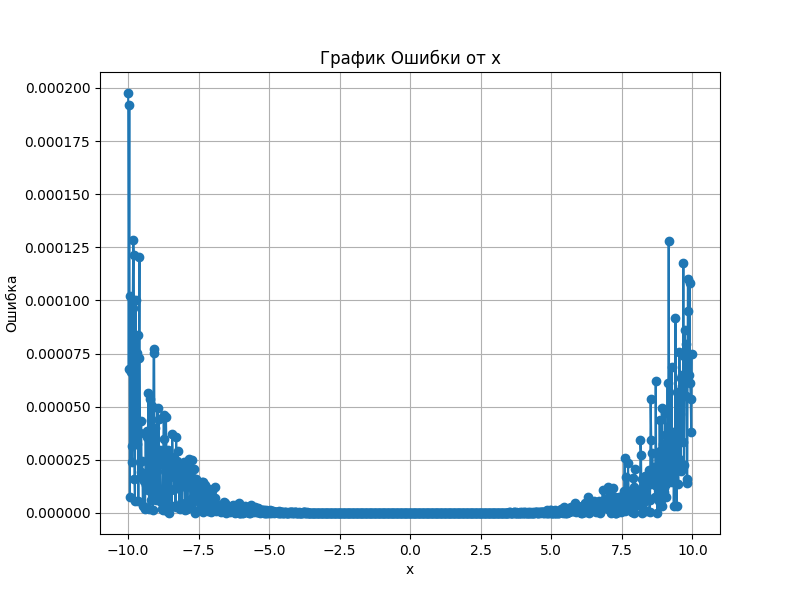
Рисунок



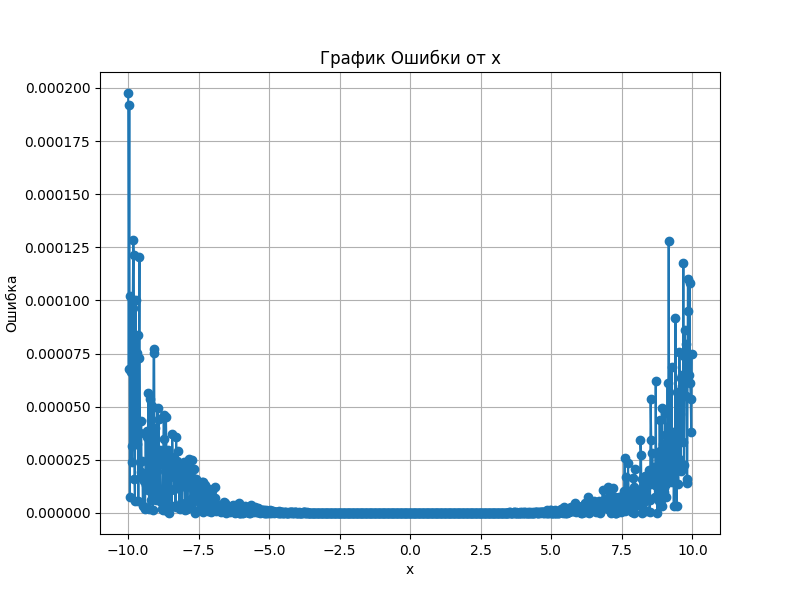
Рисунок



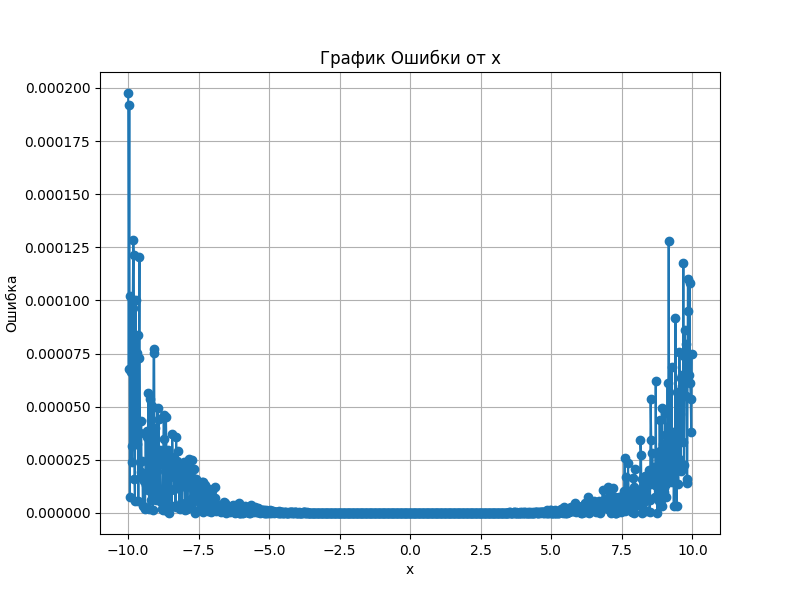
Рисунок



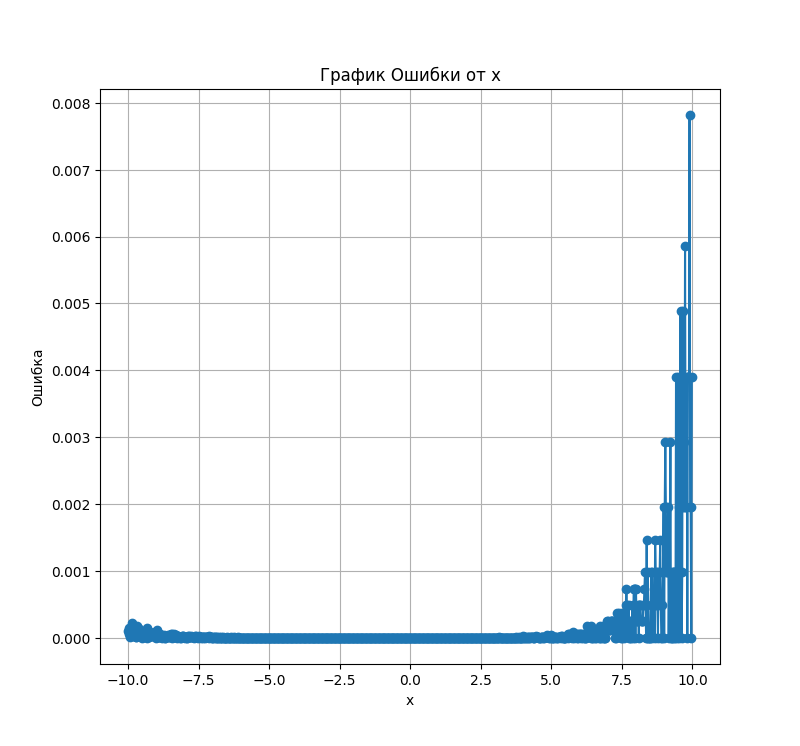
Рисунок



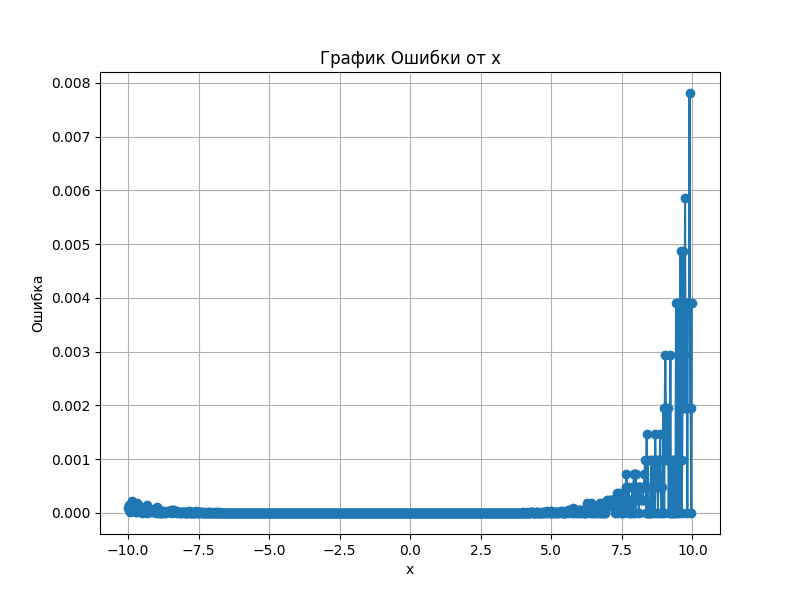
Рисунок



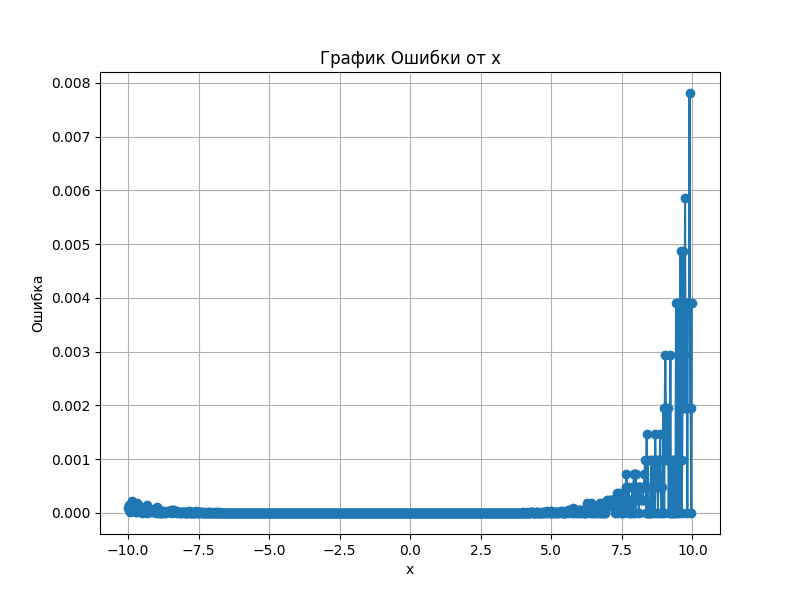
Рисунок



Рисунок



Рисунок



Рисунок

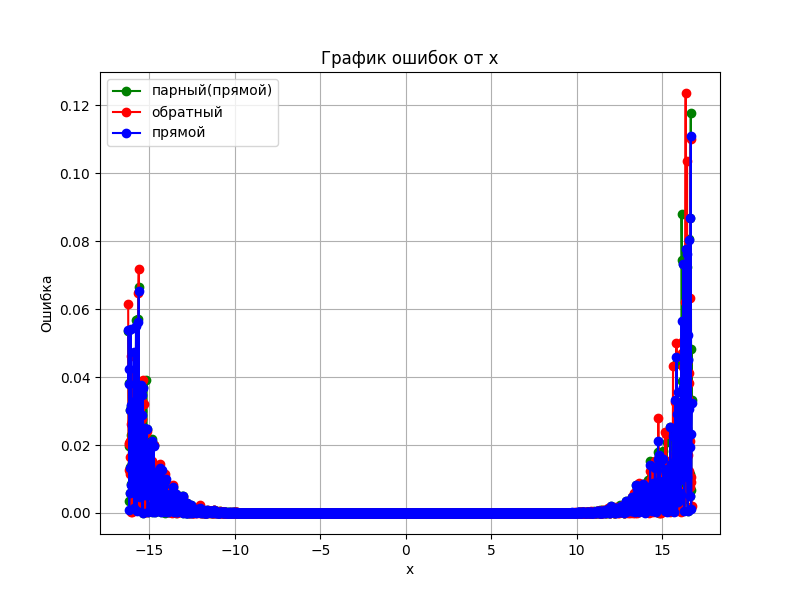
# Результаты экспериментов

В ходе эксперимента были получены следующие результаты. Данные в таблице [Table 1] представлены для x на отрезке [-1.50; 0]). (Все результаты экспериментов можно получить в каталоге “mat”)

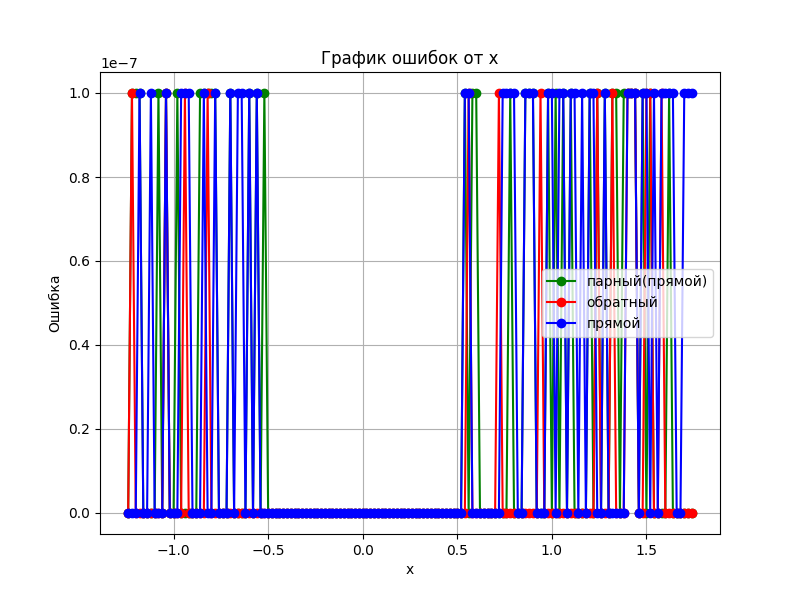
Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | Функция | Ошибка  прямого хода | Ошибка  обратного хода | Ошибка  прямого суммирования пар |
| -1.50 | Sin(x) | 0.0000001 | 0.0000000 | 0.0000000 |
| Log(1+x) | nan | nan | nan |
| Cos(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Exp(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| -1.40 | Sin(x) | 0.0000001 | 0.0000000 | 0.0000001 |
| Log(1+x) | nan | nan | nan |
| Cos(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Exp(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| -1.30 | Sin(x) | 0.0000001 | 0.0000000 | 0.0000000 |
| Log(1+x) | nan | nan | nan |
| Cos(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Exp(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| -1.20 | Sin(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Log(1+x) | nan | nan | nan |
| Cos(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Exp(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| -1.10 | Sin(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Log(1+x) | nan | nan | nan |
| Cos(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Exp(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| -1.00 | Sin(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Log(1+x) | 1.6880131 | 1.6594534 | 1.6655970 |
| Cos(x) | 0.0000001 | 0.0000000 | 0.0000000 |
| Exp(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| -0.90 | Sin(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Log(1+x) | 0.0000019 | 0.0000000 | 0.0000000 |
| Cos(x) | 0.0000000 | 0.0000001 | 0.0000000 |
| Exp(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| -0.80 | Sin(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Log(1+x) | 0.0000005 | 0.0000001 | 0.0000001 |
| Cos(x) | 0.0000001 | 0.0000000 | 0.0000000 |
| Exp(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| -0.70 | Sin(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Log(1+x) | 0.0000002 | 0.0000001 | 0.0000001 |
| Cos(x) | 0.0000000 | 0.0000001 | 0.0000001 |
| Exp(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| -0.60 | Sin(x) | 0.0000000 | 0.0000000 | 0.0000001 |
| Log(1+x) | 0.0000000 | 0.0000001 | 0.0000000 |
| Cos(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Exp(x) | 0.0000001 | 0.0000001 | 0.0000001 |
| -0.50 | Sin(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Log(1+x) | 0.0000002 | 0.0000001 | 0.0000000 |
| Cos(x) | 0.0000001 | 0.0000000 | 0.0000001 |
| Exp(x) | 0.0000001 | 0.0000000 | 0.0000001 |
| -0.40 | Sin(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Log(1+x) | 0.0000001 | 0.0000000 | 0.0000000 |
| Cos(x) | 0.0000000 | 0.0000000 | 0.0000001 |
| Exp(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| -0.30 | Sin(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Log(1+x) | 0.0000001 | 0.0000000 | 0.0000000 |
| Cos(x) | 0.0000001 | 0.0000000 | 0.0000001 |
| Exp(x) | 0.0000001 | 0.0000000 | 0.0000001 |
| -0.20 | Sin(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Log(1+x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Cos(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Exp(x) | 0.0000001 | 0.0000000 | 0.0000000 |
| -0.10 | Sin(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Log(1+x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Cos(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Exp(x) | 0.0000001 | 0.0000000 | 0.0000001 |
| 0.00 | Sin(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Log(1+x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Cos(x) | 0.0000000 | 0.0000000 | 0.0000000 |
| Exp(x) | 0.0000000 | 0.0000000 | 0.0000000 |

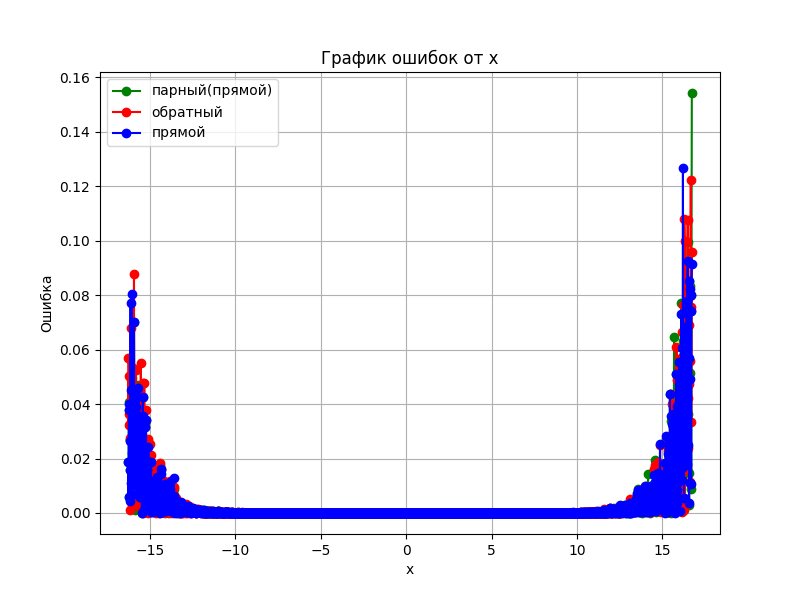
В следующих графиках показан график ошибок всех методов для каждой функции (с шагом 0.02). [Рисунок 13, Рисунок 14] – график ошибки синуса, [Рисунок 15,Рисунок 16] – график ошибки косинуса, [Рисунок 17] – график ошибки логарифма, [Рисунок 18, Рисунок 19] – график ошибки экспоненты.



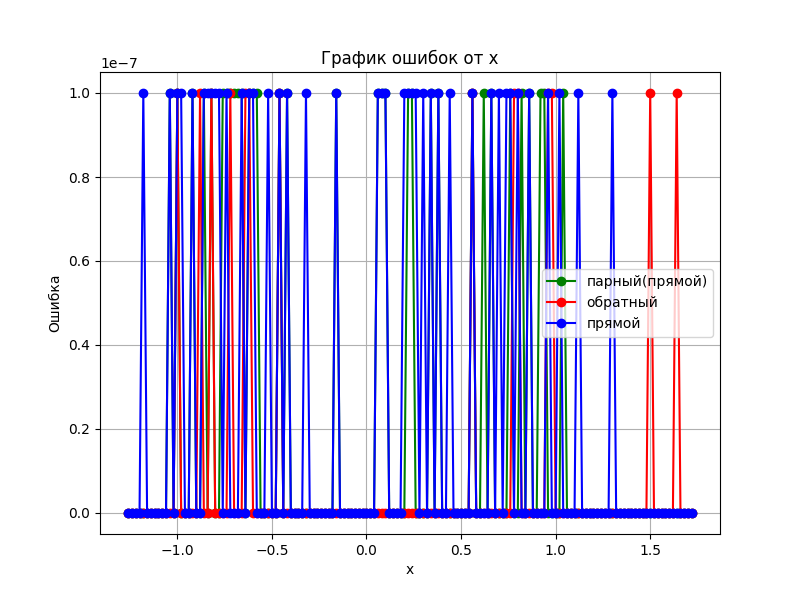
Рисунок



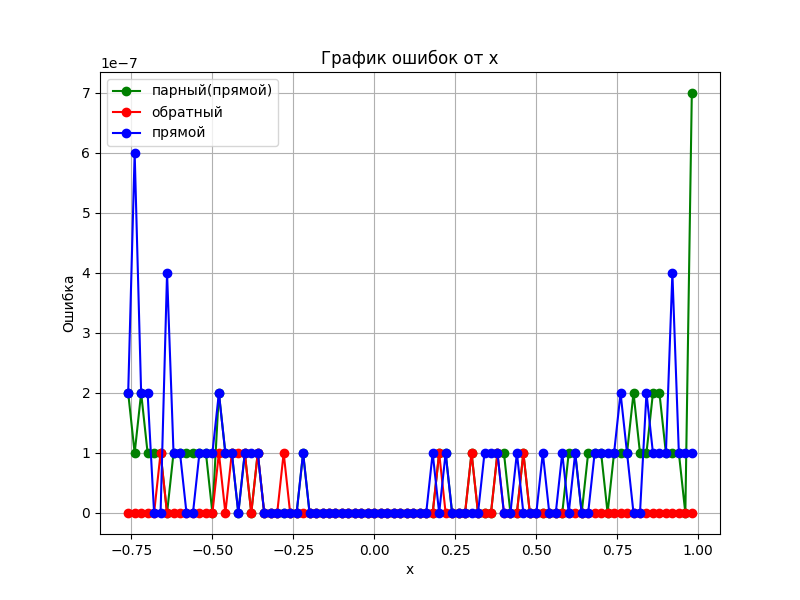
Рисунок



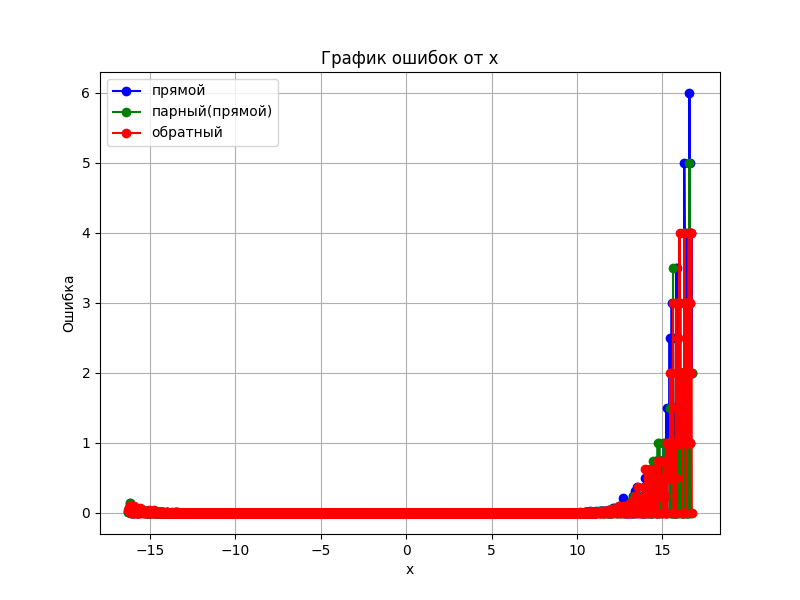
Рисунок



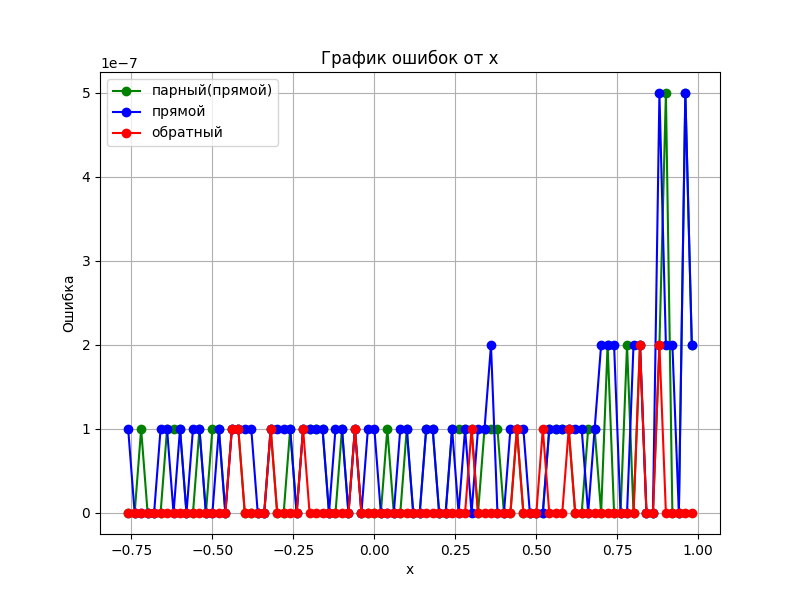
Рисунок



Рисунок



Рисунок



Рисунок

По графику [Рисунок 13] наблюдаем, что наиболее точное вычисление синуса при достаточно больших x достигается при прямом методе суммирования, но при вычислении синуса в окрестности нуля, наиболее точным методом оказался метод обратного суммирования [Рисунок 14].

При вычислении косинуса при достаточно больших х наблюдается схожий результат, что и при вычислении синуса [Рисунок 15], но при вычислении косинуса в окрестности нуля, приблизительно равные результаты дал метод обратного суммирования и метод прямого суммирования пар, в то время как метод прямого суммирования остается менее точным [Рисунок 16].

При вычислении логарифма (), хорошую точность показывает метод прямого суммирования пар, но наиболее точным остается метод обратного суммирования [Рисунок 17].

При вычислении экспоненты наиболее точным методом является метод обратного суммирования как и при больших х, так и в окрестности нуля [Рисунок 18, Рисунок 19].

# Заключение

Реализованные алгоритмы были проверены на корректность, было проведено сравнение их между собой. Самым худшим алгоритмом оказался метод прямого суммирования. Самым лучшим алгоритмом оказался метод обратного суммирования. За счёт того, что суммирование начинается с небольших значений, и с ростом результата прибавляемые значения тоже растут, погрешность суммы не велика, т.к. складываются значения с не настолько большой разностью как в методе прямого суммирования. Метод прямого суммирования пар несколько лучше метода прямого суммирования, это связано с тем, что некоторые математические функции имеют разложение с чередующимся знаком, за счет чего алгоритм намного дольше сохраняет тенденцию складывать значения с меньшей разностью значений. Из выше сказанного следует, что из предложенных алгоритмов метод обратного суммирования наиболее точный.

# Список литературы

1. Демидович Б. П. “Сборник задач и упражнений по математическому анализу. Учебное пособие для вузов”/ Б. П. Демидович. – М.: ООО «Издательство Астрель»: ООО «Издательство АСТ», 2005. (дата обращения – 20.02.2024).

# Приложение

//BEGIN-------------Вычисление множителей для последующих членов ряда---------------------------------

float sin\_p(float x, int i) //Функция вычисления множителя для следующего члена ряда для синуса

{

return -(x \* x) / (2 \* i \* (2 \* i + 1));

}

float cos\_p(float x, int i) //Функция вычисления множителя для следующего члена ряда для косинуса

{

return -(x \* x) / (2\*i\*(2\*i-1));

}

float log\_p(float x, int i) //Функция вычисления множителя для следующего члена ряда для логарифма

{

return -(x\*i) / (i + 1);

}

float exp\_p(float x, int i) //Функция вычисления множителя для следующего члена ряда для экспоненты

{

return x / i;

}

//END-------------Вычисление множителей для последующих членов ряда------------------------------------

float sum\_series\_forward(float a, float x, float(\*next)(float, int), int n) //Прямая сумма ряда

{

float res = a;

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

a = next(x, i) \* a;

res += a;

}

return res;

}

float sum\_series\_backward(float a, float x, float(\*next)(float, int), int n) //Обратная сумма ряда

{

float\* series\_mem = (float\*)malloc((n+1) \* sizeof(float)); //Члены ряда

float res = 0;

if (!series\_mem) {

printf("ERROR: can't allocate memory for series\_mem in sum\_series\_backward()\n");

getchar();

exit(1);

}

series\_mem[0] = a;

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

a = next(x, i) \* a;

series\_mem[i] = a;

}

for (int i = n; i > -1; i--) {

res += series\_mem[i];

}

free(series\_mem);

return res;

}

float sum\_series\_pair\_forward(float a, float x, float(\*next)(float, int), int n) //Прямая сумма ряда попарно

{

float res = 0.0;

float tmp1,tmp2;

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

if (!i) {

tmp1 = a;

tmp2 = (i+1 <= n) ? next(x, i+1)\*tmp1 : 0.0;

}

else {

tmp1 = next(x, i)\*tmp2;

tmp2 = (i + 1 <= n) ? next(x, i + 1)\*tmp1 : 0.0;

}

float y = tmp1 + tmp2;

res += y;

}

return res;

}