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»**

**Выполнила**:

студентка группы 3823Б1ПМ1

Абракова Виктория Вячеславовна

**Проверил**:

преподаватель каф. ВВСП,

Волокитин В.Д.

Нижний Новгород

2024

**СОДЕРЖАНИЕ**

[Постановка задачи 3](#_Toc161764017)

[Метод решения 4](#_Toc161764018)

[Руководство пользователя 5](#_Toc161764019)

[Описание программной реализации 6](#_Toc161764020)

[Подтверждение корректности 7](#_Toc161764021)

[Результаты экспериментов 8](#_Toc161764022)

[Заключение 10](#_Toc161764023)

[Список литературы 11](#_Toc161764024)

[Приложение 12](#_Toc161764025)

# Постановка задачи

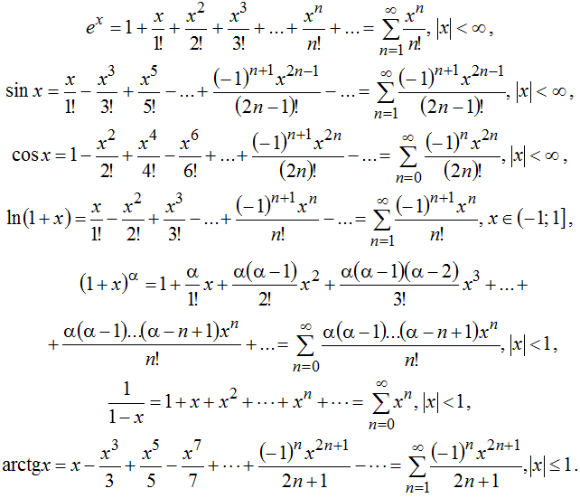
 В данной лабораторной работе требовалось написать программу, реализующую подсчёт значений таких математических функций, как: натуральный логарифм, косинус, синус, экспонента, с помощью рядов Маклорена.

Рисунок 1. Ряды Маклорена

Вычисления должны осуществляться тремя способами:

1. Прямая сумма
2. Обратная сумма
3. Попарное суммирование

В конце необходимо провести сравнение алгоритмов и выявить наиболее корректный из них.

# Метод решения

Для решения поставленной задачи, необходимо реализовать все три способа вычислений. Приведу описание их алгоритмов.

1. **Прямая сумма**

Сумма вычисляется последовательно, начина с нулевого члена ряда, заканчивая последним.

1. **Обратная сумма**

Сумма вычисляется последовательно в обратном порядке, начиная с последнего члена ряда и заканчивая первым.

1. **Попарное суммирование**

Сумма вычисляется с начала ряда. Члены ряда складываются парами, а затем добавляются в общую сумму.

# Руководство пользователя

При запуске программы на консоль выводятся значения функций ln, cos, sin, exp с аргументами от 0 до 32 с шагом 1/128, посчитанных тремя способами: прямая сумма, обратная сумма, попарное суммирование. Слева направо соответственно.

# Описание программной реализации

Программа начинает работу с функции main().

Всего в программе 8 функций:

* Функции вычисления множителей для получения последующих членов ряда: float sin\_next() - для синуса, float cos\_next() - для косинуса, float exp\_next() - для экспоненты, float ln\_next() - для логарифма.
* Функции вычисления математических функций через ряд Маклорена: float directSum() - метод прямой суммы, float inverseSum() - метод обратной суммы, float pairSum() – метод попарного суммирования.
* Функция, вызываемая при запуске программы: int main().

float sin\_next() находит множитель по формуле -x \* x / (2 \* i \* (2 \* i + 1)).

float cos\_next() находит множитель по формуле -x \* x / (2 \* i \* (2 \* i - 1)).

float exp\_next() находит множитель по формуле x / i.

float ln\_next() находит множитель по формуле -x \* i / (i + 1).

float directSum() вычисляет сумму любого ряда. Требуется передать первый член (x0), аргумент (x), указатель на функцию получения следующего члена ряда и количество элементов в ряде (n).

В цикле от до вычисляются элементы ряда, с помощью функции для получения следующих членов, и добавляются в результирующую сумму.

float inverseSum() вычисляет последний член ряда и начиная с него выполняет суммирование, находя с его помощью предшествующие члены.

Float pairSum() суммирует члены ряда в прямом порядке парами, а затем добавляет их в общую сумму и выполняет суммирование.

# Подтверждение корректности

Для подтверждения корректности реализованные математические функции сравнивались с аналогичными из стандартной библиотеки. Вблизи точки 0 значения этих функций совпадают. Можно сделать вывод, что алгоритмы реализованы корректно.

# Результаты экспериментов

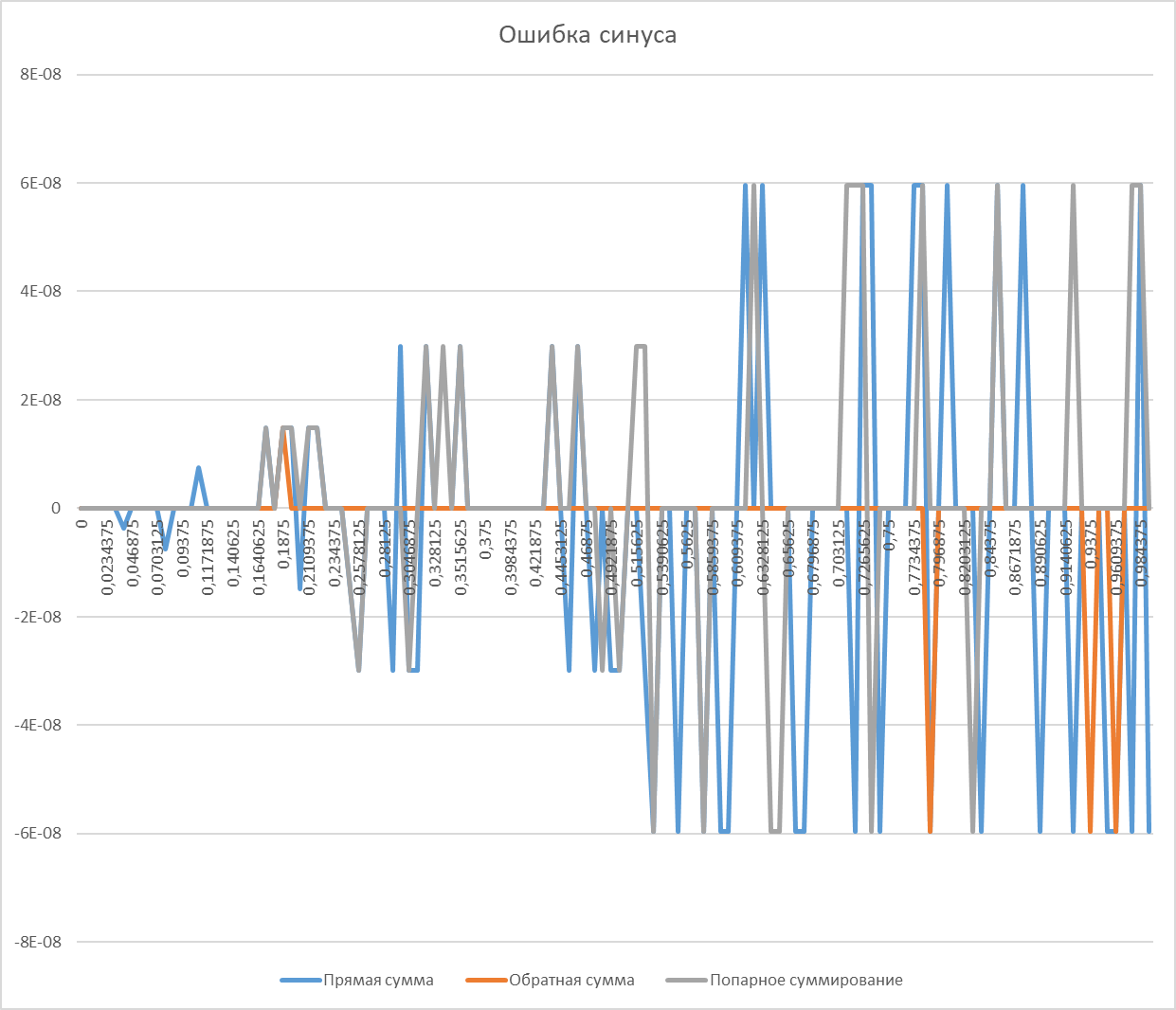
 Ошибка математических функций вычислялась как разность между значениями функций из библиотеки и реализованных функций. Ошибки каждой математической функции по отдельности представлены на Графиках 1, 2, 3, 4 в таком порядке: для синуса, косинуса, экспоненты и логарифма.

График 1. Ошибка синуса

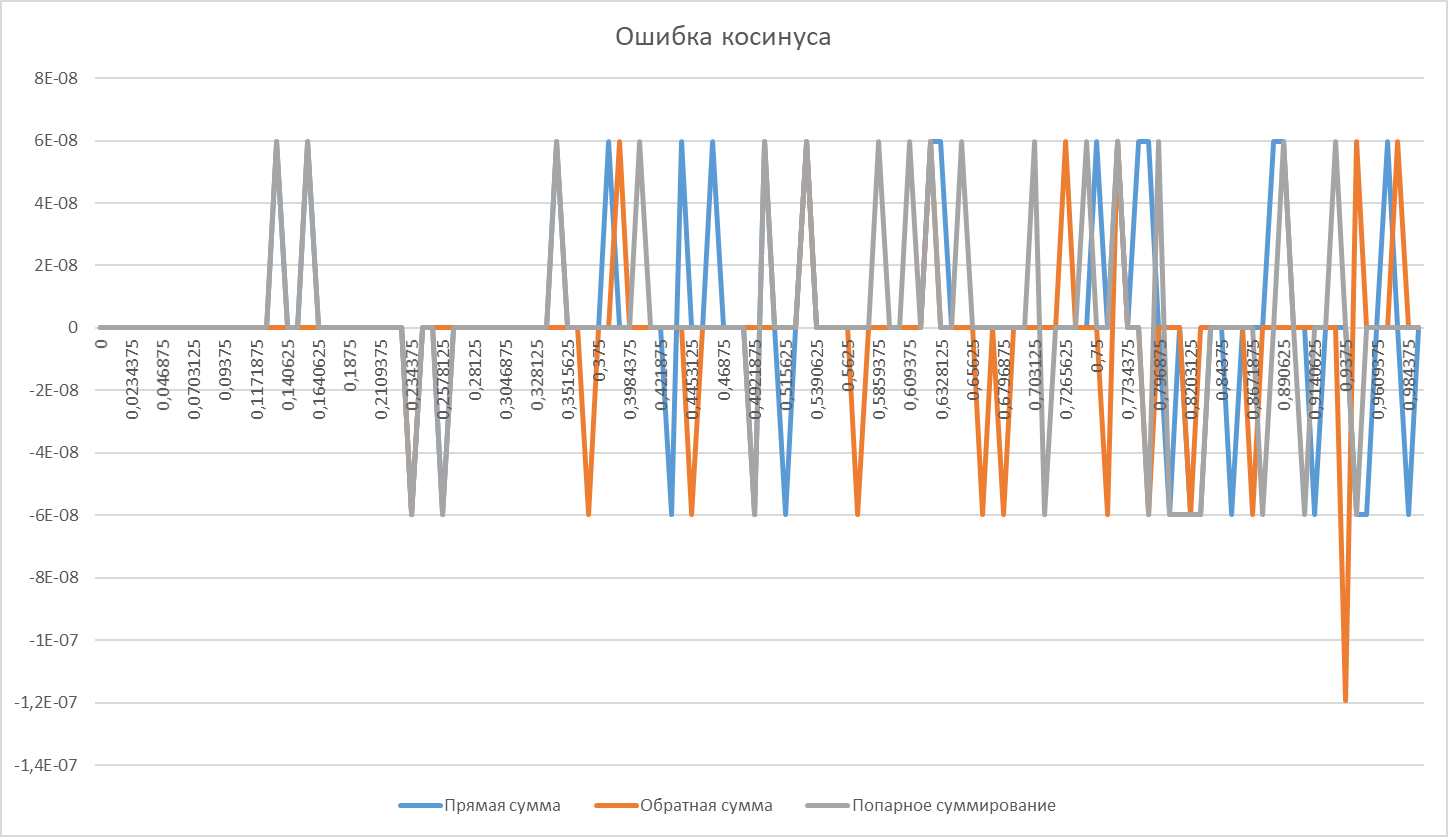


График 2. Ошибка косинуса

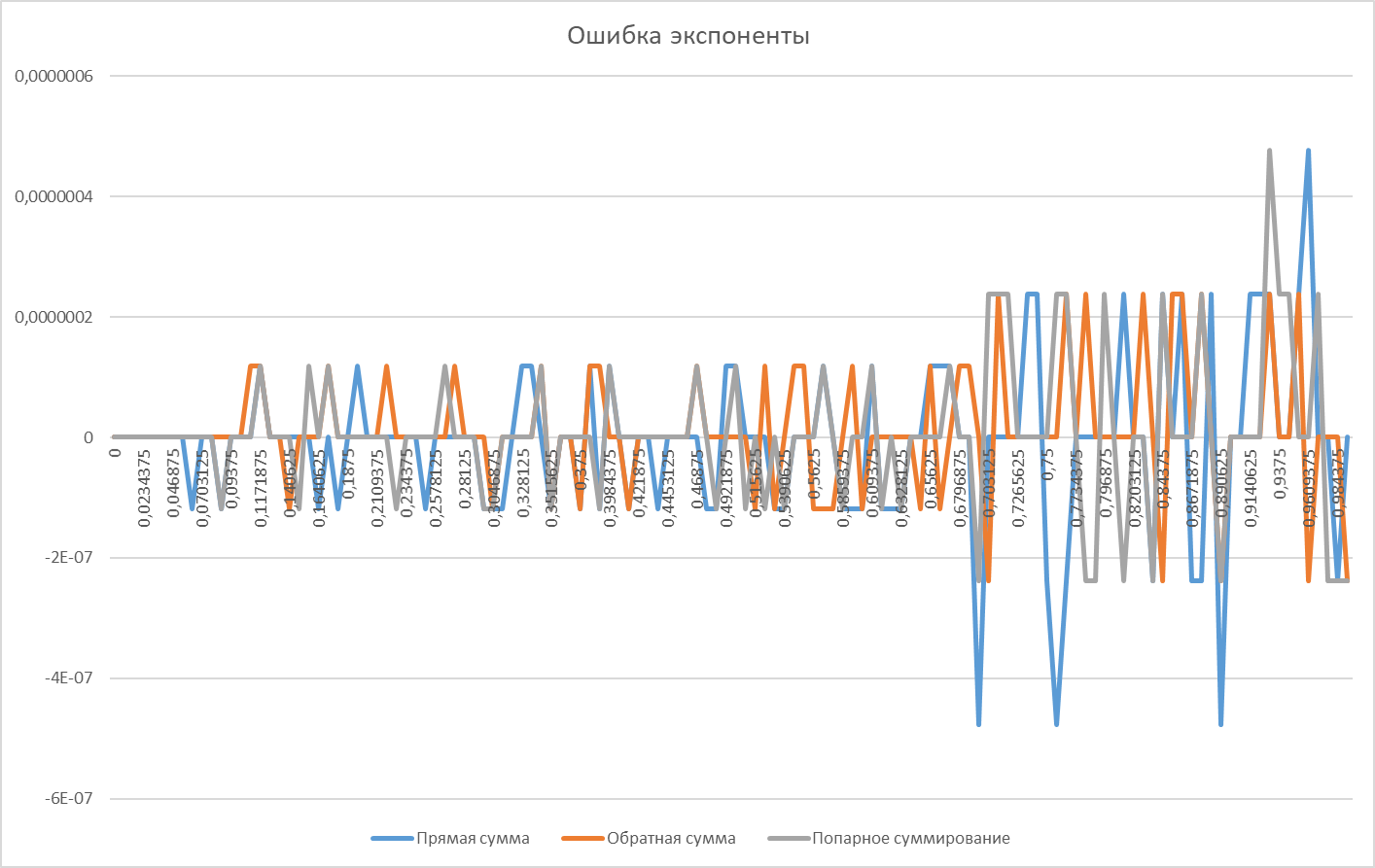
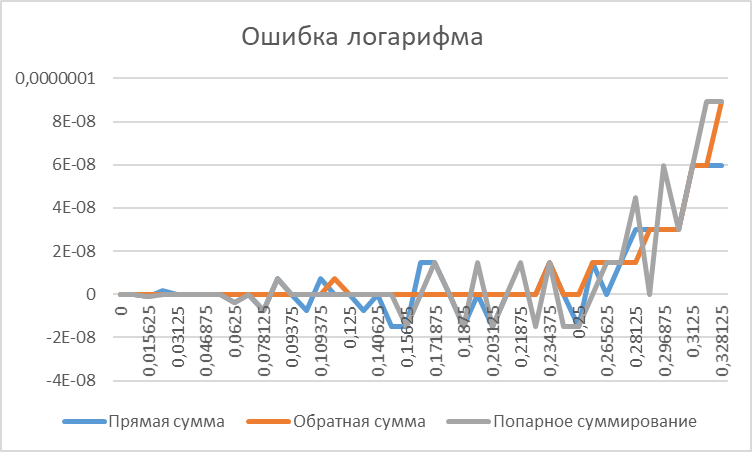


График 4. Ошибка логарифма

График 3. Ошибка экспоненты

На горизонтальной оси расположены аргументы функций, а на вертикальной значение ошибки. Видно, что при увеличении аргумента функции растёт и ошибка.

Эксперимент проводился при значениях аргумента от 0 до 1 с шагом 1/128, так как при больших значениях аргумента сложно сравнивать величину ошибок из-за их быстрого роста.

Как видно из графиков, для всех функций наиболее корректным оказался метод обратного суммирования, значения ошибок оказались меньше, чем при применении двух других методов.

Логарифм быстро расходится, поэтому он проходил проверку при достаточно малых значениях аргумента.

# Заключение

В ходе лабораторной работы была написана программа, реализующая подсчёт значений таких математических функций, как: натуральный логарифм, косинус, синус, экспонента, с помощью рядов Маклорена тремя способами: прямая сумма, обратная сумма и попарное суммирование.

В результате эксперимента было выявлено, что наиболее точно вычисления проводятся методом обратного суммирования.

# Список литературы

1. Ряды Тейлора, Бином, Тригонометрические функции, Разное, Степенные ряды: [Электронный ресурс] // Math10. URL: https://www.math10.com/ru/vysshaya-matematika/rqd-teilora.html. (Дата обращения: 4.03.2024)

# Приложение

float sin\_next(float x, int i) {

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

}

float cos\_next(float x, int i) {

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

}

float exp\_next(float x, int i) {

return x / i;

}

float ln\_next(float x, int i) {

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

}

float directSum(float x0, float x, float(\*next)(float, int), int n) {

float res, xi;

res = xi = x0;

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

{

xi \*= next(x, i);

res += xi;

}

return res;

}

float inverseSum(float x0, float x, float(\*next)(float, int), int n){

float res, xn, t;

xn = x0;

res = 0;

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

{

t = next(x, i);

if (t == 0)

{

n = i - 1;

break;

}

xn \*= t;

}

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

{

res += xn;

xn /= next(x, i);

}

return res + x0;

}

float pairSum(float x0, float x, float(\*next)(float, int), int n)

{

float res, xi, xj;

res = xi = x0;

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

{

xi \*= next(x, i);

xj = xi;

if (i + 1 <= n)

{

xi \*= next(x, ++i);

xj += xi;

}

res += xj;

}

return res;

}

float sin\_(float x, float (\*sum)(float, float, float(\*)(float, int), int))

{

return sum(x, x, sin\_next, 5);

}

float cos\_(float x, float (\*sum)(float, float, float(\*)(float, int), int))

{

return sum(1, x, cos\_next, 5);

}

float exp\_(float x, float (\*sum)(float, float, float(\*)(float, int), int))

{

return sum(1, x, exp\_next, 10);

}

float ln\_(float x, float (\*sum)(float, float, float(\*)(float, int), int))

{

return sum(x, x, ln\_next, 10);

}