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 с использованием рядов»**

**Выполнил**:

студентка группы 3824Б1ПМ1-1

Кудряшова А.А.

**Проверил**:

преподаватель каф. ВВСП,

Волокитин В.Д.

Нижний Новгород

2025

**Содержание**

[Постановка задачи 3](#_Toc26962562)

[Метод решения 4](#_Toc26962563)

[Руководство пользователя 5](#_Toc26962564)

[Описание программной реализации 6](#_Toc26962565)

[Подтверждение корректности 7](#_Toc26962566)

[Результаты экспериментов 8](#_Toc26962567)

[Заключение 9](#_Toc26962568)

[Приложение 10](#_Toc26962569)

# Постановка задачи

Требуется написать программу для вычисления значений математических функций sin(x), cos(x), eˣ и ln(1+x) с использованием их разложений в ряды Тейлора. Для каждой функции необходимо:

1. Реализовать алгоритмы суммирования (прямое,обратное,попарное) членов ряда до достижения заданной точности.
2. Сравнить результаты с данными, полученными через стандартные функции библиотеки math.h, вычислив погрешность.
3. Проанализировать эффективность методов суммирования

# Метод решения

**Прямое суммирование**:

Члены ряда суммируются в порядке их вычисления — от первого (n=0) до последнего (n=N). Преимущества : минимальный код и отсутствие дополнительных структур данных, не требует хранения всех членов ряда (O(1) памяти).Недостатки: для знакопеременных рядов (например, sin(x), cos(x)) ошибки округления могут суммироваться, если члены ряда убывают медленно, требуется больше итераций.

Описание работы:

Входные данные: x — аргумент функции, n — количество членов ряда (индексация с 0), fptr — указатель на функцию, вычисляющую очередной член ряда.

Функция с помощью цикла и вызова нужной функции находит i-е слагаемое и прибавляет его к сумме.

**Обратное суммирование**:

Члены ряда сохраняются в массив, затем суммируются в обратном порядке (от последнего к первому).Преимущества: сначала суммируются меньшие по модулю члены, что критично для знакопеременных рядов, повышение точности (для рядов с быстро убывающими членами).Недостатки: требуется O(n) памяти для хранения членов, необходимость работы с динамическими массивами.

Описание работы:

Входные данные: x — аргумент функции, n — количество членов ряда (индексация с 0), fptr — указатель на функцию, вычисляющую очередной член ряда.

Функция сохраняет все члены в массив и суммирует их в обратном порядке (от последнего к первому). Это уменьшает ошибки округления, так как сначала суммируются меньшие по модулю члены.

**Попарное суммирование**:

Члены ряда суммируются попарно. Преимущества: компенсация ошибок — для знакопеременных рядов пары членов могут частично нейтрализовать ошибки. Недостатки :усложненная логика — необходимость обработки нечетного числа членов, затраты памяти — как и в обратном суммировании (O(n)).

Описание работы:

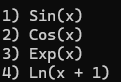
Входные данные: x — аргумент функции, n — количество членов ряда (индексация с 0), fptr — указатель на функцию, вычисляющую очередной член ряда.

Функция вычисляет пары слагаемых, добавляет их в массив и складывает эти пары как одно слагаемое. Это помогает компенсировать ошибки для знакопеременных рядов, где соседние члены имеют противоположные знаки.

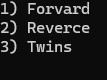
Для реализации функций, которые будут вычислять последующее слагаемое понадобятся формулы зависимости от x ( аргумент функции), n ( количество членов ряда), (предыдущее слагаемое):

# Руководство пользователя

Для работы с программой пользователь должен выбрать функцию и ввести ее номер(1-4):



Далее пользователь выбирает метод суммирования и вводит его порядковый номер(1-3):



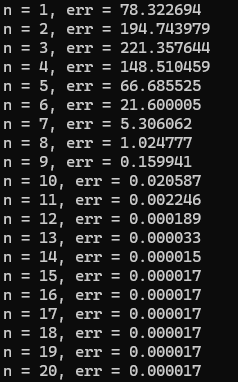
Далее пользователь выбирает точку в которой будет раскладываться функция:



Далее пользователь выбирает количество слагаемых при разложении:



После вывода пользователь получает зависимость погрешности от количества слагаемых:



# Описание программной реализации

Данная программа вычисляет значения математических функций через ряды Тейлора и сравнивает их с библиотечными реализациями из math.h

Используются библиотеки:

* stdio.h (Для ввода-вывода (printf, scanf))
* math.h (Для математических функций (sin, cos, exp, log))
* malloc.h (Для динамического выделения памяти (calloc, free))

Используются функции для членов рядов Тейлора:

* **sin\_(x, n, x\_n)**

( где x-точка в которой нужно посчитать значение,n-количество слагаемых,x\_n-предыдущее слагаемое)

* **cos\_(x, n, x\_n)**

( где x-точка в которой нужно посчитать значение,n-количество слагаемых,x\_n-предыдущее слагаемое)

* **exp\_(x, n, x\_n)**

( где x-точка в которой нужно посчитать значение,n-количество слагаемых,x\_n-предыдущее слагаемое)

* **ln\_(x, n, x\_n)**

( где x-точка в которой нужно посчитать значение,n-количество слагаемых,x\_n-предыдущее слагаемое)

Каждая из этих функций вычисляет следующее слагаемое формулы тейлора, получая на вход предыдущее значение, номер слагаемого и точку,в которой считается функция.

Используются функции суммирования:

* Прямое суммирование (forvardSum)

(x — аргумент функции, n — количество членов ряда (индексация с 0), fptr — указатель на функцию, вычисляющую очередной член ряда)

Функция с помощью цикла и вызова нужной функции находит i-е слагаемое и прибавляет его к сумме.

* Обратное суммирование (reverseSum)

(x — аргумент функции, n — количество членов ряда (индексация с 0), fptr — указатель на функцию, вычисляющую очередной член ряда)

Функция сохраняет все члены в массив и суммирует их в обратном порядке (от последнего к первому). Это уменьшает ошибки округления, так как сначала суммируются меньшие по модулю члены.

* Попарное суммирование (twinsSum)

(x — аргумент функции, n — количество членов ряда (индексация с 0), fptr — указатель на функцию, вычисляющую очередной член ряда)

Функция вычисляет пары слагаемых, добавляет их в массив и складывает эти пары как одно слагаемое. Это помогает компенсировать ошибки для знакопеременных рядов, где соседние члены имеют противоположные знаки.

# Подтверждение корректности

Для подтверждения корректности реализованных математических функций, в программе сравниваются результаты со значениями из стандартной библиотеки **math.h**. В результате работы программы выводится погрешность в виде модуля разности значений полученных мной и стандартных значений из библиотеки.

# Результаты экспериментов

Для получения результатов эксперимента рассмотрим зависимость погрешности от количества слагаемых для всех функций:

Sin(x) для х=1.5:

(рис.1) зависимость погрешности от количества слагаемых для sin(x),прямое суммирование

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,059995 |
| 2 | 0,003286 |
| 3 | 0,000104 |
| 4 | 0,000002 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |
| 12 | 0 |
| 13 | 0 |
| 14 | 0 |
| 15 | 0 |
| 16 | 0 |
| 17 | 0 |
| 18 | 0 |
| 19 | 0 |
| 20 | 0 |
| 21 | 0 |
| 22 | 0 |
| 23 | 0 |
| 24 | 0 |
| 25 | 0 |

(рис.2) зависимость погрешности от количества слагаемых для sin(x),обратное суммирование

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,059995 |
| 2 | 0,003286 |
| 3 | 0,000104 |
| 4 | 0,000002 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |
| 12 | 0 |
| 13 | 0 |
| 14 | 0 |
| 15 | 0 |
| 16 | 0 |
| 17 | 0 |
| 18 | 0 |
| 19 | 0 |
| 20 | 0 |
| 21 | 0 |
| 22 | 0 |
| 23 | 0 |
| 24 | 0 |
| 25 | 0 |

(рис.3) зависимость погрешности от количества слагаемых для sin(x),попарное суммирование

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 1,559995 |
| 2 | 0,059995 |
| 3 | 0,063385 |
| 4 | 0,000104 |
| 5 | 0,000106 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |
| 12 | 0 |
| 13 | 0 |
| 14 | 0 |
| 15 | 0 |
| 16 | 0 |
| 17 | 0 |
| 18 | 0 |
| 19 | 0 |
| 20 | 0 |
| 21 | 0 |
| 22 | 0 |
| 23 | 0 |
| 24 | 0 |
| 25 | 0 |

Cos(x)=1.5: (рис.4) зависимость погрешности от количества слагаемых для cos(x),прямое суммирование

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,195737 |
| 2 | 0,0152 |
| 3 | 0,00062 |
| 4 | 0,000016 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |
| 12 | 0 |
| 13 | 0 |
| 14 | 0 |
| 15 | 0 |
| 16 | 0 |
| 17 | 0 |
| 18 | 0 |
| 19 | 0 |
| 20 | 0 |
| 21 | 0 |
| 22 | 0 |
| 23 | 0 |
| 24 | 0 |
| 25 | 0 |

(рис.5) зависимость погрешности от количества слагаемых для cos(x), обратное суммирование

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,195737 |
| 2 | 0,0152 |
| 3 | 0,00062 |
| 4 | 0,000016 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |
| 12 | 0 |
| 13 | 0 |
| 14 | 0 |
| 15 | 0 |
| 16 | 0 |
| 17 | 0 |
| 18 | 0 |
| 19 | 0 |
| 20 | 0 |
| 21 | 0 |
| 22 | 0 |
| 23 | 0 |
| 24 | 0 |
| 25 | 0 |

(рис.6) зависимость погрешности от количества слагаемых для cos(x),попарное суммирование

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 1,195737 |
| 2 | 0,195737 |
| 3 | 0,211558 |
| 4 | 0,00062 |
| 5 | 0,000636 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |
| 12 | 0 |
| 13 | 0 |
| 14 | 0 |
| 15 | 0 |
| 16 | 0 |
| 17 | 0 |
| 18 | 0 |
| 19 | 0 |
| 20 | 0 |
| 21 | 0 |
| 22 | 0 |
| 23 | 0 |
| 24 | 0 |
| 25 | 0 |

Exp(x)=5:

(рис.7) зависимость погрешности от количества слагаемых для exp(x),прямое суммирование

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 142,4132 |
| 2 | 129,9132 |
| 3 | 109,0798 |
| 4 | 83,03816 |
| 5 | 56,9965 |
| 6 | 35,29511 |
| 7 | 19,79413 |
| 8 | 10,106 |
| 9 | 4,723706 |
| 10 | 2,032559 |
| 11 | 0,809308 |
| 12 | 0,299618 |
| 13 | 0,103589 |
| 14 | 0,033581 |
| 15 | 0,010251 |
| 16 | 0,002957 |
| 17 | 0,000806 |
| 18 | 0,00021 |
| 19 | 0,000058 |
| 20 | 0,000012 |
| 21 | 0,000003 |
| 22 | 0,000003 |
| 23 | 0,000003 |
| 24 | 0,000003 |
| 25 | 0,000003 |

(рис.8) зависимость погрешности от количества слагаемых для exp(x),обратное суммирование

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 142,413159 |
| 2 | 129,913159 |
| 3 | 109,079827 |
| 4 | 83,038159 |
| 5 | 56,996503 |
| 6 | 35,295117 |
| 7 | 19,794125 |
| 8 | 10,106 |
| 9 | 4,723721 |
| 10 | 2,032574 |
| 11 | 0,809323 |
| 12 | 0,299634 |
| 13 | 0,103604 |
| 14 | 0,033597 |
| 15 | 0,010251 |
| 16 | 0,002957 |
| 17 | 0,000821 |
| 18 | 0,000226 |
| 19 | 0,000058 |
| 20 | 0,000027 |
| 21 | 0,000012 |
| 22 | 0,000012 |
| 23 | 0,000012 |
| 24 | 0,000012 |
| 25 | 0,000012 |

(рис.9) зависимость погрешности от количества слагаемых для exp(x),обратное суммирование

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 143,4132 |
| 2 | 142,4132 |
| 3 | 121,5798 |
| 4 | 109,0798 |
| 5 | 83,03816 |
| 6 | 56,9965 |
| 7 | 41,49551 |
| 8 | 19,79413 |
| 9 | 14,41183 |
| 10 | 4,723721 |
| 11 | 3,50047 |
| 12 | 0,809323 |
| 13 | 0,613293 |
| 14 | 0,103604 |
| 15 | 0,080273 |
| 16 | 0,010251 |
| 17 | 0,008099 |
| 18 | 0,000806 |
| 19 | 0,000653 |
| 20 | 0,000058 |
| 21 | 0,000043 |
| 22 | 0,000012 |
| 23 | 0,000012 |
| 24 | 0,000012 |
| 25 | 0,000012 |

Ln(1+x)=0.2:

(рис.10) зависимость погрешности от количества слагаемых для ln(1+x),прямое суммирование

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,002322 |
| 2 | 0,000345 |
| 3 | 0,000055 |
| 4 | 0,000009 |
| 5 | 0,000002 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |
| 12 | 0 |
| 13 | 0 |
| 14 | 0 |
| 15 | 0 |
| 16 | 0 |
| 17 | 0 |
| 18 | 0 |
| 19 | 0 |
| 20 | 0 |
| 21 | 0 |
| 22 | 0 |
| 23 | 0 |
| 24 | 0 |
| 25 | 0 |

(рис.11) зависимость погрешности от количества слагаемых для ln(1+x),обратное суммирование

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,002322 |
| 2 | 0,000345 |
| 3 | 0,000055 |
| 4 | 0,000009 |
| 5 | 0,000002 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |
| 12 | 0 |
| 13 | 0 |
| 14 | 0 |
| 15 | 0 |
| 16 | 0 |
| 17 | 0 |
| 18 | 0 |
| 19 | 0 |
| 20 | 0 |
| 21 | 0 |
| 22 | 0 |
| 23 | 0 |
| 24 | 0 |
| 25 | 0 |

(рис.12) зависимость погрешности от количества слагаемых для ln(1+x),попарное суммирование

|  |  |
| --- | --- |
| Число слагаемых | Погрешность |
| 1 | 0,202322 |
| 2 | 0,002322 |
| 3 | 0,002722 |
| 4 | 0,000055 |
| 5 | 0,000066 |
| 6 | 0,000002 |
| 7 | 0,000002 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |
| 12 | 0 |
| 13 | 0 |
| 14 | 0 |
| 15 | 0 |
| 16 | 0 |
| 17 | 0 |
| 18 | 0 |
| 19 | 0 |
| 20 | 0 |
| 21 | 0 |
| 22 | 0 |
| 23 | 0 |
| 24 | 0 |
| 25 | 0 |

Выводы:

**Рассмотрев рисунки 1-6 можно понять, что для функций sin(x) и cos(x)**:

* Все методы суммирования (прямое, обратное, попарное) обеспечивают быстрое уменьшение погрешности до нуля при n≥5.
* Попарное суммирование демонстрирует всплески погрешности на малых n, что связано с особенностями компенсации знакопеременных членов.
* Для этих функций обратное суммирование не дает значительного преимущества перед прямым из-за быстрой сходимости ряда

**Рассмотрев рисунки 6-7 можно понять, что для функции exp(x)**:

* Ряд Тейлора для exp(x) сходится медленно при x=5 Погрешность уменьшается постепенно, достигая 3×10−6 при n=25.
* Попарное суммирование показывает нестабильность и более высокую погрешность по сравнению с прямым и обратным методами. Это связано с монотонным характером ряда, где парная компенсация неэффективна.
* Обратное суммирование демонстрирует схожие результаты с прямым, но требует больше памяти.

Функция ln(1+x): Для x=0.2,все методы обеспечивают нулевую погрешность уже при n≥6 что объясняется быстрой сходимостью ряда вблизи

x=0Попарное суммирование снова показывает всплески погрешности на малых n, но для больших n n все методы эквивалентны.

# Заключение

**Прямое суммирование** наиболее универсально. Оно подходит для всех типов рядов, но для знакопеременных функций (например, sin(x)) может накапливать ошибки округления на больших n. **Обратное суммирование** обеспечивает сравнимую точность с прямым методом, но требует O(n) памяти. Его стоит использовать для знакопеременных рядов с быстрой сходимостью. **Попарное суммирование** менее предсказуемо. Оно может быть полезно для знакопеременных рядов, но требует осторожности из-за возможных всплесков погрешности на малых n.

Для sin(*x*), cos(*x*),ln(1+*x*) все методы приемлемы, но предпочтительно прямое или обратное суммирование. Для exp при больших *x* рекомендуется использовать прямое суммирование с увеличенным числом итераций. Попарное суммирование следует применять только для специфических задач, где критична компенсация ошибок.

# Приложение

#include <stdio.h>

#include <math.h>

#include <malloc.h>

float sin\_(float x, int n, float x\_n) {

if (n == 0)

return x;

return -x\_n \* (x \* x) / (4 \* n \* n + 2 \* n);

}

float cos\_(float x, int n, float x\_n) {

if (n == 0)

return 1.0;

return -x\_n \* (x \* x) / (4 \* n \* n - 2 \* n);

}

float exp\_(float x, int n, float x\_n) {

if (n == 0)

return 1.0;

return x\_n \* (x / n);

}

float ln\_(float x, int n, float x\_n) {

if (n == 0)

return x;

return -x\_n \* (n \* x) / (n + 1);

}

float forvardSum(float x, int n, float(\*fptr)(float, int, float)) {

float sum = 0, prev = 0;

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

prev = fptr(x, i, prev);

sum += prev;

}

return sum;

}

float reverseSum(float x, int n, float(\*fptr)(float, int, float)) {

float sum = 0, prev = 0;

float\* t\_arr = (float\*)calloc(n + 1, sizeof(float));

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

t\_arr[i] = prev = fptr(x, i, prev);

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

sum += t\_arr[i];

free(t\_arr);

return sum;

}

float twinsSum(float x, int n, float(\*fptr)(float, int, float)) {

float sum = 0, prev = 0;

float\* t\_arr = (float\*)calloc(n + 1, sizeof(float));

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

t\_arr[i] = prev = fptr(x, i, prev);

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

sum += t\_arr[i] + t\_arr[i + 1];

if (n % 2 != 0)

sum += t\_arr[n];

return sum;

}

void main() {

printf("1) Sin(x) \n2) Cos(x) \n3) Exp(x) \n4) Ln(x + 1) \n");

int func\_c = 0;

float(\*fptr)(float, int, float) = NULL;

double(\*eFptr)(double) = NULL;

scanf\_s("%d", &func\_c);

switch (func\_c) {

case 1:

fptr = sin\_;

eFptr = sin;

break;

case 2:

fptr = cos\_;

eFptr = cos;

break;

case 3:

fptr = exp\_;

eFptr = exp;

break;

case 4:

fptr = ln\_;

eFptr = log1p;

break;

default:

break;

}

printf("1) Forvard \n2) Reverce \n3) Twins \n");

int sum\_c = 0;

scanf\_s("%d", &sum\_c);

printf("Enter x: \n");

float x = 0.0;

scanf\_s("%f", &x);

printf("Enter max n: \n");

int n = 0;

scanf\_s("%d", &n);

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

switch (sum\_c) {

case 1:

printf("n = %d, err = %f \n", i, fabs(eFptr(x) - forvardSum(x, i, fptr)));

break;

case 2:

printf("n = %d, err = %f \n", i, fabs(eFptr(x) - reverseSum(x, i, fptr)));

break;

case 3:

printf("n = %d, err = %f \n", i, fabs(eFptr(x) - twinsSum(x, i, fptr)));

break;

default:

break;

}

}

}