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**

**«Подсчет sin(x), cos(x), exp(x), ln(x+1)»**

**Выполнила**:

студентка группы 3821Б1ПМ2

Анисимова Ю. Д.

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2022

**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 10](#_Toc26962568)

[Приложение 11](#_Toc26962569)

# Постановка задачи

Задачей моей лабораторной работы является реализация вычисления значения функций sin, cos, exp, ln(1+x) в точке с помощью ряда Маклорена. Необходимо реализовать методы прямого, попарного и обратного суммирования элементов. Нужно описать реализацию и алгоритмы работы программы. Необходимо подтвердить корректность реализации вычисления данных функций. Провести эксперименты по замеру точности различных методов суммирования и результаты сравнить с библиотечным значением.

# Метод решения

Ряд Тейлора – разложение функции в бесконечную сумму степенных функций. С помощью этого ряда можно найти приближённое значение функции в окрестности заданной точки.

Разложения по формуле Тейлора для заданных функций выглядят так:

Задача выполняется с помощью прямого, обратного и попарного суммирования.

**Прямое суммирование**

Слагаемые складываются в порядке увеличения их порядкового номера (каждое слагаемое находится с использованием предыдущего)

**Обратное суммирование**

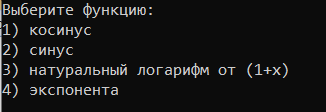
Слагаемые складываются в порядке уменьшения их порядкового номера (каждое слагаемое находится с использованием предыдущего)

**Попарное суммирование**

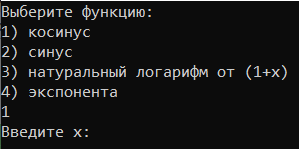
Попарное суммирование представляет собой метод суммирования последовательности чисел конечной точности с плавающей запятой. Попарное суммирование последовательности из n чисел x n работает посредством рекурсивно разбиения последовательности на две половины, суммирования каждой половины и сложения двух.

# Руководство пользователя

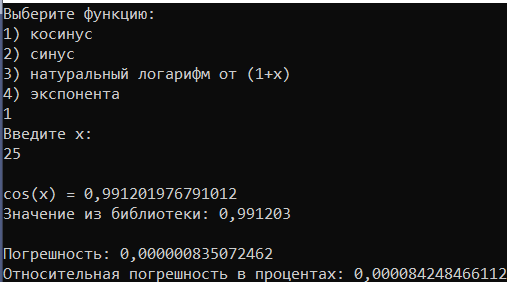
При запуске программы появляется окно ввода, где предлагается выбрать функцию для подсчета.



Далее предлагается ввести число x, от которого будет считаться выбранная функция:



В результате работы программы будет выводится значение выбранной функции, которое посчитала программа, значение этой же функции из библиотеки math.h, погрешность вычислений и относительная погрешность, выраженная в %.



# Описание программной реализации

Функции, написанные ниже, являются функциями суммирования элементов ряда: прямое, попарное и обратное (принимают массив элементов ряда и производят суммирование).

float direct\_summ(float\* arr, int m)

float revers\_summ(float\* arr, int m)

float paired\_summ(float\* arr, int m)

Функции, написанные ниже, вычисляют элементы ряда для каждой математической функции (эти функции возвращают численное значение ряда):

float cos\_1(float pred, float n, int i)

float sin\_1(float pred, float n, int i)

float ln\_1(float pred, float n, int i)

float exp\_1(float pred, float n, int i)

Функция, создающая массив элементов ряда Маклорена.

void made\_array(float\* arr, int m, option result, float p)

# Подтверждение корректности

Для подтверждения корректности в программе вычисляется относительная и абсолютная погрешности.

Абсолютная погрешность вычисляет разницу между библиотечным значением и значением, которое посчитала программа. Можно наглядно увидеть, насколько точно работает программа.

Относительная погрешность вычисляется в процентах. Абсолютная погрешность делится на значение, которое посчитала программа и все это умножается на 100.

# Результаты экспериментов

SIN(x):

|  |  |  |  |
| --- | --- | --- | --- |
| Значение x | Относительная погрешность прямого суммирования в % | Относительная погрешность попарного суммирования в % | Относительная погрешность обратного суммирования в % |
| -5 | 0,00007656 | 0,00005791 | 0,00010142 |
| -1 | 0,00000333 | 0,00000333 | 0,00000333 |
| -0,5 | 0,00000394 | 0,00000394 | 0,00000227 |
| 0,5 | 0,00000394 | 0,00000394 | 0,00000227 |
| 1 | 0,00000333 | 0,00000333 | 0,00000333 |
| 5 | 0,00007656 | 0,00005791 | 0,00010142 |
| 10 | 0,01822379 | 0,01826761 | 0,03935851 |
| 12 | 0,01789846 | 0,01258864 | 0,09997840 |
| 17 | 6,75431617 | 6,94515817 | 5,29812762 |

COS(x):

|  |  |  |  |
| --- | --- | --- | --- |
| Значение x | Относительная погрешность прямого суммирования в % | Относительная погрешность попарного суммирования в % | Относительная погрешность обратного суммирования в % |
| -5 | 0,00002640 | 0,00006815 | 0,00012096 |
| -1 | 0,00000541 | 0,00000541 | 0,00000541 |
| -0,5 | 0,00000544 | 0,00000544 | 0,00000135 |
| 0,5 | 0,00000544 | 0,00000544 | 0,00000135 |
| 1 | 0,00000541 | 0,00000541 | 0,00000541 |
| 5 | 0,00002640 | 0,00006815 | 0,00012096 |
| 10 | 0,01370092 | 0,01370802 | 0,00110616 |
| 12 | 0,01287557 | 0,00564973 | 0,01231952 |
| 16 | 0,27341345 | 2,63304121 | 0,54997651 |

Можно сделать вывод, что лучший способ сложения для этих функций – попарное суммирование, но на небольших значениях X, лучше работает обратное суммирование.

EXP(x):

|  |  |  |  |
| --- | --- | --- | --- |
| Значение x | Относительная погрешность прямого суммирования в % | Относительная погрешность попарного суммирования в % | Относительная погрешность обратного суммирования в % |
| -5 | 0,00706997 | 0,00706306 | 0,01062224 |
| -1 | 0,00001372 | 0,00000561 | 0,00000249 |
| -0,5 | 0,00001092 | 0,00001092 | 0,00000109 |
| 0,5 | 0,00000404 | 0,00000404 | 0,00000319 |
| 1 | 0,00000573 | 0,00000573 | 0,00000304 |
| 5 | 0,00002267 | 0,00000211 | 0,00000211 |
| 10 | 0,00001342 | 0,00001342 | 0,00000455 |
| 12 | 0,00001585 | 0,00002545 | 0,09997840 |
| 20 | 0,00000424 | 0,00001744 | 0,00002403 |
| 30 | 0,00002297 | 0,00003279 | 0,00000335 |
| 40 | 0,00001311 | 0,00000581 | 0,00000149 |
| 50 | 0,00002308 | 0,00001223 | 0,00001223 |
| 80 | 1,71081789 | 1,71081789 | 1,71080895 |

Для функции экспоненты лучший способ подсчета – обратное суммирование. При небольших X значения функции вычисляются довольно точно.

LN(1+X):

|  |  |  |  |
| --- | --- | --- | --- |
| Значение x | Относительная погрешность прямого суммирования в % | Относительная погрешность попарного суммирования в % | Относительная погрешность обратного суммирования в % |
| -0,9 | 0,00013322 | 0,00007109 | 0,00009180 |
| -0,7 | 0,00000004 | 0,00000559 | 0,00000431 |
| -0,5 | 0,00001092 | 0,00000027 | 0,00000027 |
| -0.25 | 0,00000032 | 0,00000479 | 0,00000557 |
| 0,25 | 0,00000826 | 0,00000159 | 0,00000159 |
| 0,5 | 0,00001028 | 0,00000293 | 0,00000442 |
| 0,7 | 0,00000748 | 0,00000748 | 0,00000375 |
| 0,9 | 0,00004716 | 0,00001001 | 0,00001930 |
| 1,0 | 0,71775212 | 0,71775212 | 0,71773492 |

Для функции натурального логарифма лучший способ подсчета – попарное суммирование. Значения функции при всех значениях вычисляются довольно точно.

# Заключение

В ходе лабораторной работы я реализовала вычисления значений функций sin, cos, exp, ln(1+x) в точке с помощью ряда Маклорена. Реализовала методы прямого, попарного и обратного суммирования элементов. Описала реализацию и алгоритмы работы программы. Подтвердила корректность реализации вычисления данных функций. Провела эксперименты по замеру точности различных методов суммирования и результаты сравнила с библиотечным значением.

# Приложение

float direct\_summ(float\* arr, int m)

{

float summ = 0;

int i;

for (i = 0; i < m; i++)

{

summ += arr[i];

}

return summ;

}

float revers\_summ(float\* arr, int m)

{

float summ = 0;

int i;

for (i = 1; i <= m; i++)

{

summ += arr[m - i];

}

return summ;

}

float paired\_summ(float\* arr, int m)

{

int i;

float summ = 0;

for (i = 0; i < m - 1; i += 2)

arr[i] += arr[i + 1];

if (m % 2 == 1)

summ += arr[i + 2];

for (i = 0; i < m; i += 2)

summ += arr[i];

return summ;

}

float cos\_1(float pred, float n, int i)

{

i = 2 \* i;

return (pred \* (-1.f) \* n \* n) / (i \* (i - 1.f));

}

float sin\_1(float pred, float n, int i)

{

i = 2 \* i + 1;

return (pred \* (-1.f) \* n \* n) / (i \* (i - 1.f));

}

float ln\_1(float pred, float n, int i)

{

return (pred \* (-1.f) \* n \* i) / (i + 1.f);

}

float exp\_1(float pred, float n, int i)

{

return (pred \* n) / ((float)i);

}

void made\_array(float\* arr, int m, option result, float p)

{

int i;

for (i = 1; i < m; i++)

{

arr[i] = result(arr[i - 1], p, i);

}

}