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**

**«Методы суммирования»**

**Выполнил**:

студентка группы 3821Б1ПМ-2

Нагорная Е.А.

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2022

**Содержание**

[Постановка задачи 3](#_Toc89867396)

[Методы решения 4](#_Toc89867397)

[Прямое суммирование 4](#_Toc89867398)

[Обратное суммирование 4](#_Toc89867399)

[Попарное суммирование 4](#_Toc89867400)

[Руководство пользователя 5](#_Toc89867402)

[Описание программной реализации 6](#_Toc89867403)

[Прямое суммирование 6](#_Toc89867404)

[Обратное суммирование 6](#_Toc89867405)

Попарное суммирование [7](#_Toc89867406)

[Результаты экспериментов 8](#_Toc89867409)

Прямое суммирование [8](#_Toc89867410)

[Обратное суммирование 8](#_Toc89867411)

[Попарное суммирование 8](#_Toc89867412)

[Заключение 9](#_Toc89867414)

Приложения  [10](#_Toc89867410)

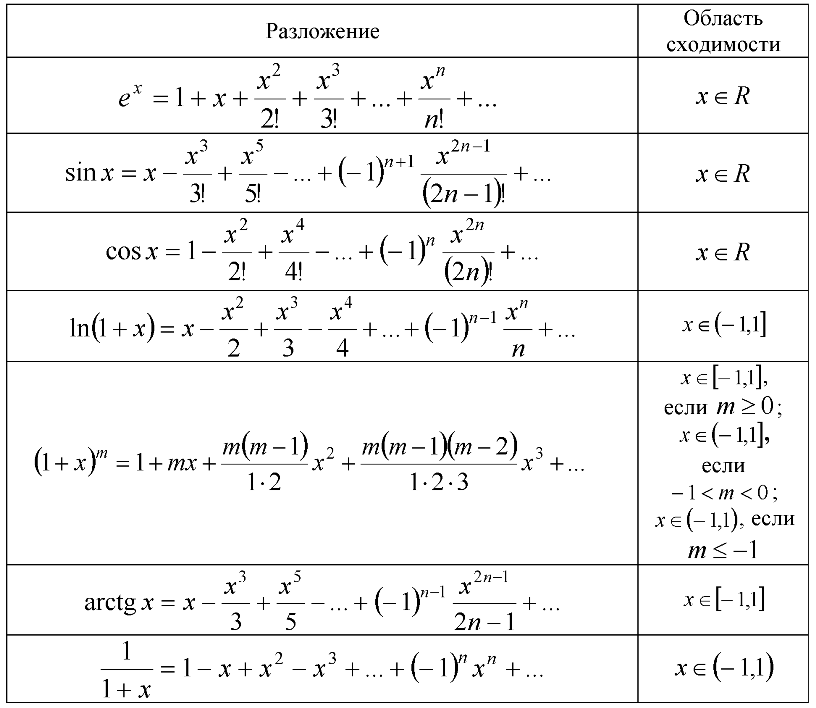
# Постановка задачи

Целью лабораторной работы являлось реализовать на языке программирования Си подсчёт элементарных математических функций разными видами суммирования элементов рядов Маклорена. Подсчёт нужно реализовать для данных типа float. Нужно описать программную реализацию и алгоритмы работы данных видов суммирования. Провести эксперименты для вычисления абсолютной и относительной погрешности вычислений, описать способ проведения экспериментов и сделать вывод по полученным результатам.

# Метод решения

Все функции вычисляются при использовании рядов Маклорена (рядов Тейлора в окрестности нуля).Многочленом Тейлора функции {\displaystyle f(x)}f(x) вещественной переменной {\displaystyle x}x называется конечная сумма, {\displaystyle f(x)=\sum \_{n=0}^{k}{\frac {f^{(n)}(a)}{n!}}(x-a)^{n}=f(a)+f'(a)(x-a)+{\frac {f^{(2)}(a)}{2!}}(x-a)^{2}+\ldots +{\frac {f^{(k)}(a)}{k!}}(x-a)^{k}}используемая в приближённых вычислениях, как обобщение следствия теоремы Лагранжа о среднем значении дифференцируемой функции:

Элементы заполняемого массива – слагаемые каждого из рядов:



**Прямое суммирование**

Последовательное сложение элементов в порядке возрастания номера слагаемого. Например, 1+2+3+4+…+(n-1)+n.

**Обратное суммирование**

Последовательное сложение элементов в порядке убывания номера слагаемого. Например, n+(n-1)+(n-2)+(n-3)+…+2+1.

**Попарное суммирование**

Последовательное сложение сумм соседних пар элементов в порядке возрастания. Например, (1+2)+(3+4)+…+((n-1)+n).

# Руководство пользователя

В консоль выводится число, для которого вычисляются элементарные математические функции, а также абсолютные и относительные погрешности при подсчёте каждой из функций разными видами суммирования. Вывод осуществляется в следующем формате:

“число;

АП1, ОП1, АП2, ОП2, АП3, ОП3; -для синуса числа

АП1, ОП1, АП2, ОП2, АП3, ОП3; -для косинуса числа

АП1, ОП1, АП2, ОП2, АП3, ОП3; -для экспоненты числа

АП1, ОП1, АП2, ОП2, АП3, ОП3; -для натурального логарифма числа, увеличенного на 1

”, где АП - абсолютная погрешность, ОП – относительная погрешность, 1-прямое суммирование, 2-обратное суммирование, 3-папарное суммирование.

Для каждого числа вывод начинается с новой строки.

# Описание программной реализации

Массив, задающий число х для вычисления математических функций:

for (j = -10; j < 10; j += 0.1) { x = j; }

Массив, вычисляющий слагаемые для формулы

*А) Синуса х:*

A[0] = x; k = 1;

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

{

k = k + 2;

A[i] = A[i - 1] \* x \* x / k / (k - 1) \* (-1);

}

*Б) Косинуса х:*

A[0] = 1; k = 0;

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

{

k = k + 2;

A[i] = A[i - 1] \* x \* x / k / (k - 1) \* (-1);

}

*В) Экспоненты х:*

A[0] = 1; k = 0;

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

{

k = k + 1;

A[i] = A[i - 1] \* x / k;

}

*Г) Натурального логарифма х+1:*

A[0] = x; k = 1;

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

{

k = k + 1;

A[i] = A[i - 1] \* x / k \* (-1);

}

Массив, подсчитывающий значения функций F методом

*А) Прямого суммирования*

F1 = A[0];

for (i = 1; i < n; i++) { F1 += A[i]; }

*Б) Обратного суммирования*

F2 = A[n-1];

for (i = n-2; i >= 0; i--) { F2 += A[i]; }

*В) Попарного суммирования*

F3 = 0;

for (i = 0; i < n; i += 2) { F3 += (A[i] + A[i + 1]); }

Вычисление функций для проверки:

prov = sin(x);

prov = cos(x);

prov = exp(x);

prov = log(x+1);

Вывод заданного числа:

printf("%.1lf \n", x);

Вывод абсолютной /относительной погрешности при подсчёте функции F:

printf("%.30lf; %.30lf; ", (F1 - prov), (F1 - prov) / prov);

printf("%.30lf; %.30lf; ", (F2 - prov), (F2 - prov) / prov);

printf("%.30lf; %.30lf \n",(F3 - prov), (F3 - prov) / prov);

# Результаты экспериментов

Число х изменялось от -10 до 9,5 с шагом 0,5. Все вычисляемые нами элементарные функции определены на промежутке, кроме ln (x+1), она считалась на (-1;1] с шагом 0,1.

По полученным данным о погрешностях подсчёта функций таблицы заполнялись таблицы EXEL. Через функцию МИН вычислялись минимальные значения относительных погрешностей всех функций для каждого из видов суммирования. Для каждой из элементарных математических функций сравнивалось количество полученных значений.

По полученным данным можно сделать вывод о том, что для более точного подсчёта функции sin (x) и cos (x) сложно выделить наилучший вид суммирования (по моим исходным данным это попарное для синуса и прямое для косинуса). Для функций exp (x) и ln (x+1) анализ однозначно указывает на наилучший вид суммирования (обратное суммирование минимализирует относительные погрешности при подсчёте обеих функций).

# 

# Заключение

В ходе лабораторной работы на языке программирования Си были реализованы прямое, обратное и попарное суммирование для массива, элементы которого имеют тип данных float.

Были описаны алгоритмы работы данных суммирований и их программная реализация, а также проведены эксперименты для замера и подтверждения их абсолютной и относительной погрешностей.

**Приложения**

