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

Карпенко Р.В.

**Проверил:**

преподаватель каф. ВВСП

Волокитин В.Д.

Нижний Новгород

2024

**Содержание**

[**Постановка задачи** 3](#_Toc166083867)

[**Метод решения** 4](#_Toc166083868)

[**Руководство пользователя** 5](#_Toc166083869)

[**Описание программной реализации** 6](#_Toc166083870)

[**Подтверждение корректности** 8](#_Toc166083871)

[**Результаты экспериментов** 9](#_Toc166083872)

[**Заключение** 14](#_Toc166083873)

[**Список использованной литературы** 15](#_Toc166083874)

[**Приложение** 16](#_Toc166083875)

**Постановка задачи**

В лабораторной работе необходимо реализовать программу, вычисляющую элементарные функции(sin(x), cos(x), exp, log(x + 1)) с помощью ряда Маклорена (рис.1)

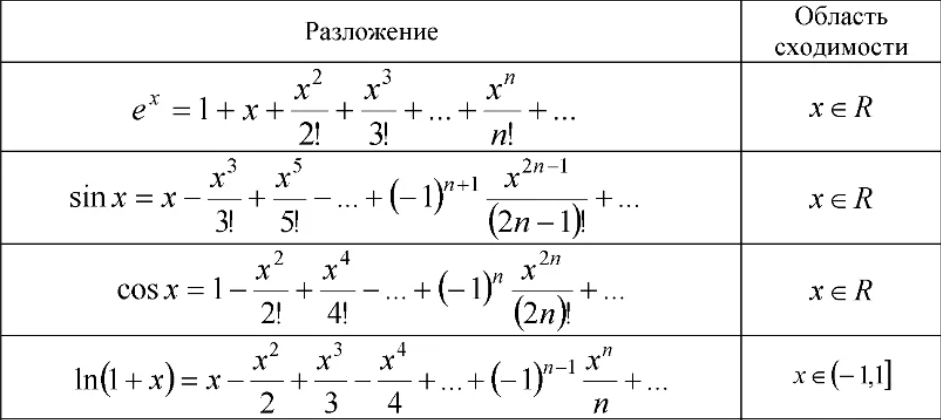


Рисунок 1. Разложение элементарных функций с помощью ряда Маклорена.

Кроме того, реализовать и сравнить 3 способа суммирования рядов:

1. Прямое суммирование рядов
2. Обратное суммирование рядов
3. Прямое попарное суммирование рядов

Выявить наиболее эффективный способ с наименьшей погрешностью.

**Метод решения**

В лабораторной работе необходимо реализовать 3 метода суммирования ряда:

1. **Прямое суммирование:** сумма ряда вычисляется последовательно от нулевого члена до заданного включительно.
2. **Обратное суммирование:** алгоритм вычисляет члены ряда методом прямого суммирования.В итоге вычисленные значения записываются в массив. Итоговая сумма происходит по массиву, начиная с последнего элемента по первый включительно.
3. **Прямое попарное суммирование:** заводятся 2 переменные для хранения двух соседних членов ряда и после вычисления прибавляются к переменной результата (общей сумме ряда)

**Руководство пользователя**

Взаимодействие с пользователем происходит через консоль. На вход принимается значения x, после чего вычисляется элементарные функции: sin(x), cos(x), exp, ln (x + 1)

**Описание программной реализации**

Всего в программе 8 функций:

● Функции вычисления множителей членов ряда: float sin\_next() –для синуса,

float cos\_next() – для косинуса, float exp\_next() – для экспоненты, float log\_next() – для логарифма

● Функции для вычисления элементарных функций с помощью ряда Маклорена: float sum\_forward() – прямое суммирование, float sum\_backward() – обратное суммирование, float sum\_pair() – прямое попарное суммирование

● int main() – функция для запуска программы

Алгоритмы функций:

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

float sum\_forward() принимает следующие 4 аргумента:

●float x0 – первый член ряда

●float x – значение функции

●float(\*next)(float,int) – функция вычисления множителей для получения последующих членов ряда.

●int n – номер члена до которого вычисляется ряд

Функция запускает цикл от 1 до n включительно, вычисляет элементы ряда с помощью функции для получения следующего члена ряда и прибавляет к результату (общую сумму)

float sum\_backward() принимает следующие 4 аргумента:

●float x0 – первый член ряда

●float x – значение функции

●float(\*next)(float,int) – функция вычисления множителей для получения последующих членов ряда.

●int n – номер члена до которого вычисляется ряд

Создается массив mas для хранения членов ряда, в начало массива записывается нулевой член. Суммирование происходит по получившемуся массиву

float sum\_pair() принимает следующие 4 аргумента:

●float x0 – первый член ряда

●float x – значение функции

●float(\*next)(float,int) – функция вычисления множителей для получения последующих членов ряда.

Заводятся две переменные xi, xj для хранения соседних членов ряда. В итоге переменные суммируются парами, результат суммы прибавляется к результату (общей сумме)

**Подтверждение корректности**

Для подтверждения корректности программы реализованные функции сравнивались с функциями из стандартной библиотеки math.h. Функции дают верный результат вблизи точки 0. Следовательно, алгоритмы были реализованы корректно.

**Результаты экспериментов**

В ходе эксперимента были получены следующие результаты: на графиках изображена разность между реализованными математическими функциями и функциями из библиотеки math.h при увеличении аргумента.

(Рис.1. Ошибка синуса при вычислении методом попарной суммы)

**(**Рис.2. Ошибка синуса при вычислении методом прямой суммы**)**

(Рис.3. Ошибка синуса при вычислении методом обратной суммы)

(Рис.4. Ошибка косинуса при вычислении методом прямой суммы)

(Рис.5. Ошибка косинуса при вычислении методом попарной суммы)

(Рис.6. Ошибка косинуса при вычислении методом обратной суммы)

(Рис.7. Ошибка экспоненты при вычислении методом прямой суммы)

(Рис.8. Ошибка экспоненты при вычислении методом попарной суммы)

(Рис.9. Ошибка экспоненты при вычислении методом обратной суммы)

(Рис.10. Ошибка логарифма при вычислении методом прямой суммы)

(Рис.11. Ошибка логарифма при вычислении методом попарной суммы)

(Рис.12. Ошибка логарифма при вычислении методом обратной суммы)

Горизонтальная ось определяет значение аргумента, вертикальная – значение ошибки. Видно, что при увеличении аргумента функции растет и ошибка.

Как видно по графикам, для всех функций наиболее точным оказался метод обратного суммирования, обычная прямая сумма является худшей по точности.

**Заключение**

В ходе лабораторной работы реализованные функции (sin(x), cos(x), exp, ln(1 + x)) были проверены на корректность, также проведено сравнение их между собой. Элементарные функции вычислялись с помощью ряда Маклорена, а суммирование рядов реализовывалось тремя разными способами: прямое суммирование, обратное и попарное суммирование.

По результатам эксперимента самым лучшим алгоритмом оказался метод обратного суммирования. Причина в том, что начало суммирование начинается с маленьких значений, и с ростом результата прибавляемые значения тоже растут, поэтому погрешность суммы значительно меньше, чем при методе прямого суммирования, где суммирование начинается с наибольших значений.

**Список использованной литературы**

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

**Приложение**

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

{

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

}

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

{

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

}

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

{

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

}

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

{

return x / i;

}

float sum\_forward(float x0, float x, float(\*next)(float, int), int n) //Прямая сумма

{

float res = x0;

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

x0 = next(x, i) \* x0;

res += x0;

}

return res;

}

float sum\_backward(float x0, float x, float(\*next)(float, int), int n) //Обратная сумма

{

float\* mas = (float\*)malloc((n + 1) \* sizeof(float)); //массив элементов ряда

float res = 0;

mas[0] = x0;

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

x0 = next(x, i) \* x0;

mas[i] = x0;

}

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

res += mas[i];

}

free(mas);

return res;

}

float sum\_pair(float x0, float x, float(\*next)(float, int), int n) //попарная сумма

{

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;

}