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**

**«Математические фунцкии»**

**Выполнил**:

студент группы 3823Б1ПМ1

Калинин Б. А.

**Проверил**:

преподаватель каф. ВВСП,

Волокитин В.Д.

Нижний Новгород

2024

**Содержание**

[Постановка задачи 3](#_Toc161780233)

[Метод решения 4](#_Toc161780234)

[Руководство пользователя 5](#_Toc161780235)

[Описание программной реализации 6](#_Toc161780236)

[Подтверждение корректности 8](#_Toc161780237)

[Результаты экспериментов 9](#_Toc161780238)

[Заключение 12](#_Toc161780239)

[Источники 13](#_Toc161780240)

[Приложение 14](#_Toc161780241)

# Постановка задачи

В данной лабораторной работе были поставлены следующие задачи:

* Реализовать на C 4 математические функции (sin, cos, exp, ln) для типа данных float, используя их разложение по формуле Тейлора (Маклорена).
* Сравнить точность 4 различных методов суммирования членов этих разложений (прямая сумма, обратная сумма, попарное суммирование, обратное попарное суммирование).

# Метод решения

Для вычисления Математических функций используется их аппроксимация рядом Маклорена, в котором коэффициент при i-ой степени x будет равен значению i-ой производной разлагаемой функции в точке x=0. Ниже приведены разложения функций, которые использовались в работе:

Для суммирования будут использоваться всего 4 различных метода. Ниже дана их характеристика:

* Прямая сумма – при её вычислении просто складываются все члены разложения слева направо.
* Обратная сумма – в данном методе члены складываются в обратном порядке, что позволяет сократить погрешность при прибавлении очень малых чисел типа float к очень большим.
* Попарное суммирование – в данном методе члены сначала суммируются попарно, и только потом пары складываются между собой. Этот подход нацелен на сокращение погрешности в рядах с чередующимися знаками.
* Обратное попарное суммирование – попарное суммирование в обратном порядке.

# Руководство пользователя

Программная реализация всех методов и математических функций представлена в файле 2lab.cpp.

При запуске программы для пользователя будет выведено меню, в котором будет предложено выбрать математическую функцию. После выбора пользователь должен будет аналогичным образом выбрать метод суммирования (или же запустить сразу все). Наконец от пользователя требуется ввести x и количество членов ряда (при выборе логарифма следует учитывать его область сходимости [-1, 1]). После этого на экран будет выведено полученное значение нужной функции, и его разность со значением соответствующей функции библиотеки math.h.

# Описание программной реализации

Программная реализация всех методов и математических функций представлена в файле 2lab.cpp.

Функцией, в которой вычисляется значение математической функции, является функция taylor с сигнатурой float taylor(float x, void(\*series\_function)(float\*, int, float), float(\*summation\_function)(float\*, int), int number\_of\_elements). Она принимает x для которого нужно посчитать значение, указатель на функцию, которая вычисляет элементы ряда, Маклорена, указатель на функцию, которая суммирует данный ряд, а также количество членов ряда. Она реализована следующим образом: создаётся массив mas, содержащий number\_of\_elements элементов со значениями членов. Он заполняется функцией series\_function и суммируется функцией summation\_function, функция возвращает результат суммирования.

4 следующие функции используются для заполнения массива с членами ряда, они принимают одни и те же аргументы (указатель на массив значений членов mas, степень до которой нужно разложить функцию n, и x) и ничего не возвращают, каждая отвечает за свою математическую функцию:

* void series\_sin(float\* mas, int n, float x) – в mas[0] присваивает x и вычисляет все последующие i-ые члены как –mas[i-1]\*x\*x/(2\*i\*(2\*i+1)).
* void series\_cos(float\* mas, int n, float x) – в mas[0] присваивает 1 и вычисляет все последующие i-ые члены как –mas[i-1]\*x\*x/(i\*2(2\*i-1)).
* void series\_exp(float\* mas, int n, float x) – в mas[0] присваивает 1 и вычисляет все последующие i-ые члены как mas[i-1]\*x/i для всех i начиная с 1.
* void series\_lnplus1(float\* mas, int n, float x) – в mas[0] присваивает 0 и вычисляет все последующие i-ые члены как var/i (еcли i- нечётное) или –var/i (если i - чётное), где var соответствует x в степени i ( изначально var = x, и после каждой итерации var\*=x), для всех i начиная с 1,.

Ещё 4 функции отвечают за суммирование массива и так же принимают одни и те же аргументы (mas – указатель на массив значений, n – количество членов ряда) и возвращают сумму элементов массива:

* float direct\_sum(float\* mas, int n) - прямая сумма, объявляет sum, обнуляет его и прибавляет к sum все значения подряд из массива mas начиная с mas[0], после возвращает сумму.
* float reverse\_sum(float\* mas, int n) - обратная сумма, делает всё то же самое, что и прямая но суммирует в обратном порядке начиная с mas[n].
* float paired\_direct\_sum(float\* mas, int n) – попарное суммирование, кроме суммы объявляет буфер buf и счётчик сложенных элементов k. При суммировании складывает элементы в прямом порядке в буфер, если элемент массива не нулевой увеличивает счётчик, при k=2 обнуляет счётчик и прибавляет значение в буфере к основной сумме. После основного цикла прибавляет то, что осталось в буфере, и возвращает сумму.
* float paired\_reverse\_sum(float\* mas, int n) - обратное попарное суммирование, то же самое что и обычное, но складывает элементы в обратном порядке.

В функции main реализовано меню выбора функции, метода суммирования, значения x и количества членов ряда в виде цикла while(1) с возможностью выхода при вводе 0, или любых других необозначенных для пользователя символов.

# Подтверждение корректности

Для проверки точности реализованных математических функций в программе используются их реализации из библиотеки math.h, а именно sinf, cosf, expf, log1pf.

# Результаты экспериментов

В результате экспериментов были получены данные для сравнения методов суммирования. Тесты проводились на относительно малых значениях x (так как при больших x становится большой погрешность, связанная с ограниченностью членов разложения).

Ниже приведены графики, показывающие погрешность методов суммирования на 4 исследуемых математических функциях.

Рисунок Погрешность sin(x) [0; 4]

Рисунок Погрешность cos(x) [0;4]

Рисунок Погрешность exp(x) [0; 4]

Рисунок Погрешность ln(x+1) [0; 1]

Так как графики очень хаотичны и не дают сделать явных выводов, ниже приведена таблица средней погрешности, составленная на меньшем отрезке [-1; 1] ([-0.8; 0.8] для ln(x)), но на большем количестве значений (10000).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Прямая сумма | Обратная сумма | Попарная сумма | Обратная попарная сумма |
| sin(x) | 3,08E-08 | 3,58E-09 | 2,40E-08 | 1,89E-08 |
| cos(x) | 4,52E-08 | 1,26E-08 | 3,83E-08 | 3,17E-08 |
| exp(x) | 1,22E-07 | 3,44E-08 | 9,10E-08 | 4,41E-08 |
| ln(1+x) | 1,02E-07 | 1,62E-08 | 6,99E-08 | 2,08E-08 |

По полученным данным видно, что в среднем наиболее точным методом является обратная сумма, среди 2 попарных методов более точным является обратная сумма, а прямая сумма оказалась наименее точным методом.

# Заключение

В лабораторной работе были реализованы все необходимые математические функции и методы суммирования. Их точность была показана на графиках, и были посчитаны средние значения погрешности.

По результатам лабораторной работы можно сделать следующие выводы:

* Разложение в ряд Маклорена может использоваться для вычисления математических функций только при очень малых значениях x (из-за точности), также оно не может использоваться для ln(x+1) вне промежутка [-1; 1] (из-за несходимости ряда).
* Среди 4 рассмотренных методов суммирования наиболее точным является обратная суммы, так как они предотвращают погрешность от суммирования больших чисел с малыми.
* В знакочередующихся рядах попарная сумма работает точнее, чем обычная прямая, при этом обратная попарная сумма работает менее точно, чем обычная обратная.

# Источники

[1] Б.П. Демидович. СБОРНИК. задач и упражнений. математическому анализу. (с. 132 - 133)

# Приложение

Основной код:

//series functions

void series\_sin(float\* mas, int n, float x) {

int i;

memset(mas, 0, sizeof(float) \* (n+1));

mas[1] = x;

for (i = 3; i <= n; i += 2)

mas[i] = -(mas[i - 2] \* x \* x) / (i \* (i - 1));

return;

}

void series\_cos(float\* mas, int n, float x) {

int i;

memset(mas, 0, sizeof(float) \* (n + 1));

mas[0] = 1;

for (i = 2; i <= n; i += 2)

mas[i] = -(mas[i - 2] \* x \* x) / (i \* (i - 1));

}

void series\_exp(float\* mas, int n, float x) {

int i;

mas[0] = 1;

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

mas[i] = mas[i - 1] \* x / i;

}

}

void series\_lnplus1(float\* mas, int n, float x) {

int i;

mas[0] = 0;

float var = x;

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

mas[i] = i % 2 ? var / i : -var / i;

var \*= x;

}

}

//summation functions

float direct\_sum(float\* mas, int n) {

float sum = 0;

int i;

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

sum += mas[i];

}

return sum;

}

float reverse\_sum(float\* mas, int n) {

float sum = 0;

int i;

for (i = n; i >= 0; i--) {

sum += mas[i];

}

return sum;

}

float paired\_direct\_sum(float\* mas, int n) {

float sum = 0;

float buf = 0;

int k = 0;

int i;

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

if (mas[i] != 0) {

buf += mas[i];

k++;

if (k == 2) {

sum += buf;

k = 0;

buf = 0;

}

}

}

sum += buf;

return sum;

}

float paired\_reverse\_sum(float\* mas, int n) {

float sum = 0;

float buf = 0;

int k = 0;

int i;

for (i = n; i >= 0; i--) {

if (mas[i] != 0) {

buf += mas[i];

k++;

if (k == 2) {

sum += buf;

k = 0;

buf = 0;

}

}

}

sum += buf;

return sum;

}

//the taylor func itself

float taylor(float x, void(\*series\_function)(float\*, int, float), float(\*summation\_function)(float\*, int), int highest\_degree) {

float\* mas;

float res;

mas = (float\*)malloc((highest\_degree+1) \* sizeof(float));

series\_function(mas, highest\_degree, x);

res = summation\_function(mas, highest\_degree);

free(mas);

return res;

}