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 3823Б1ПМ1-1

Загрядсков М.А.

**Проверил**:

преподаватель каф. ВВСП,

Волокитин В.Д.

Нижний Новгород

2024

**Содержание**

[Постановка задачи 3](#_Toc152527359)

[Метод решения 4](#_Toc152527360)

[Руководство пользователя 7](#_Toc152527361)

[Описание программной реализации 8](#_Toc152527362)

[Подтверждение корректности 9](#_Toc152527363)

[Результаты экспериментов 10](#_Toc152527364)

[Заключение 16](#_Toc152527365)

[Приложение 17](#_Toc152527366)

[Источники 18](#_Toc152527367)

# Постановка задачи

Требовалось:

1. Реализовать функции суммирования рядов Маклорена тремя различными способами: прямое суммирование, попарное суммирование и обратное суммирование. Каждая из функций принимает аргумент, нулевой член, указатель на функцию для вычисления следующего множителя, на который необходимо домножить текущий член, чтобы получить следующий, а также количество слагаемых. Также в каждую из функций был добавлен указатель на функцию, преобразующую аргумент таким образом, чтобы данный ряд Маклорена сходился для любого значения аргумента.
2. С помощью данных функций реализовать функции основных математических функций (синуса, косинуса, экспоненты и натурального логарифма);
3. Написать программу с понятным интерфейсом, позволяющую получить значение функции по заданному аргументу;
4. Проверить программу на корректность;
5. Исходный код программ загрузить в указанный репозиторий;
6. Провести ряд экспериментов, в ходе которых выяснить, какой из способов суммирования наиболее точно вычисляет значение заданной функции, сравнивая полученные значения с значениями тех же функций, реализованных в библиотеке cmath.

# Метод решения

Требовалось найти формулу для следующего множителя для данных математических функций. Приведу их ряды Маклорена:

Следующий множитель требовалось найти в виде , где -ое слагаемое.

Для решения поставленной задачи, необходимо реализовать все 3 вида суммирования. Приведу их описания:

1. **Прямое суммирование:** суммирование слагаемых в прямом порядке следования их в формуле ряда. Таким образом, сложение происходит от больших по значению слагаемых к меньшим.
2. **Попарное суммирование:** суммирование слагаемых парами, т.е. в общую сумму добавляется не , а . Разница между частичными суммами (парами слагаемых) меньше, чем между самими слагаемыми, поэтому теоретически должна достигаться большая точность.
3. **Обратное суммирование:** суммирование слагаемых в обратном порядке следования, т.е. сначала вычисляется последний, -ный член суммы, затем к нему прибавляются слагаемые . Если достаточно велико, то последний член суммы может быть достаточно мал, чтобы в типе данных float стать тождественным нулём. В этом случае будет использован последний ненулевой член.

Также для сравнения с другими способами был реализован способ, минимизирующий ошибки float и исключающий ошибку тождественного нуля для обратного суммирования: обратное суммирование, где каждый член суммы сохранялся в ячейку массива, который затем суммировался с последней ячейки к первой.

# Руководство пользователя

Взаимодействие с программой осуществляется через консоль. Последовательность и формат ввода описаны в программе. При некорректном вводе метода программа выведет “Method not found”, при некорректном вводе функции программа завершит работу. В ходе взаимодействия с программой можно выбрать одну из четырех функций, выбрать метод суммирования, ввести количество слагаемых (итераций) или запустить функцию без дополнительных начальных данных. В конце будет выведено значение, а также погрешность в форме разности значения функции из cmath и функции, которую использует программа.

# Описание программной реализации

В папке проекта доступны следующие файлы:

1. Заголовочный файл с описанием функций, используемых при вычислении рядов sumseries.hpp.
2. Файл реализации этих функций sumseries.cpp.
3. Файл решения Visual Studio MathLabProg.sln.
4. Файл проекта Visual Studio MathLabProg.vcxproj.
5. Файл описания собственных математических функций \_math.hpp.
6. Файл реализации этих функций \_math.cpp.
7. Файл MathLabProg.cpp – код программы, позволяющей взаимодействовать с пользователем через консоль.
8. Файл MathLabExperiments.cpp – код программы, тестирующей функции на корректность и записывающей погрешности в файл.
9. Файл Отчет\_лабораторная\_МатФункции\_Загрядсков\_Максим\_3823Б1ПМ1-1.docx – данный файл с текстом отчёта.

# Подтверждение корректности

Для подтверждения корректности в программе значения функций сравнивались с значениями тех же функций при тех же аргументах, реализованных в библиотеке cmath. В результате при программа относительно точно вычисляла значения данных функций.

# Результаты экспериментов

Эксперименты проводились следующим образом: выбирался отдельный метод суммирования, затем происходило вычисление всех 4-х функций, после чего в файл записывалась разность значения функции из cmath и функции из \_math для одного аргумента. Вычисления производились для типа данных float, аргумент принимал значения от 0 до 64 с шагом в .

После первых нескольких экспериментов было обнаружено, что для слишком маленьких значений – количества итераций – точность сильно снижается, а при больших значениях функция обратного суммирования вычисляла ряд с большой ошибкой. Поэтому эмпирически была получена функция зависимости количества итераций от значения переменной , где – некоторые константы.

Сперва были получены точечные диаграммы разности значения функции из библиотеки cmath и значения функции суммирования для каждой функции и каждого типа суммирования на разных масштабах. Приведу некоторые из них:

*Рис. 1. Диаграммы разности значений прямой и попарной суммы для функции синуса и значений функции синуса из cmath.*

Как можно заметить здесь, попарная сумма синуса начинает расходиться немного раньше прямой суммы. Как можно будет увидеть позже, прямая сумма для синуса точнее, чем попарная.

*Рис. 2. Диаграммы разности значений обратной суммы функции синуса и обратной суммы функции синуса с массивом и значений функции синуса из cmath.*

На данных диаграммах видно, что обратная сумма синуса без массива начинает раньше расходиться и обладает большей погрешностью вблизи 0. Это может быть связано с тем, что вблизи нуля значения синуса малы, поэтому погрешность получения последнего члена и погрешность деления для получения старших членов в сумме наиболее заметны.

*Рис. 4. Диаграммы разности значений прямой суммы функции натурального логарифма и обратной суммы функции натурального логарифма с массивом и значений функции натурального логарифма из cmath.*

На данных диаграммах видно, что для обоих типов суммирования характерна большая погрешность при маленьких значениях аргумента. Но для обратной суммы погрешность гораздо больше и доходит до значений . Я связываю это с неверным получением последнего члена для суммы.

*Рис. 4. Диаграммы разности значений прямой суммы функции синуса, косинуса и натурального логарифма и значений функций синуса, косинуса и натурального логарифма из cmath.*

Как видно из диаграмм, все суммы принимают наиболее точные значения вблизи нулей функций.

Остальные диаграммы смотрите в приложении.

Следующим этапом в исследовании было сравнение точности суммирований. Для этого среди 4х значений разности выбиралось ближайшее к нулю, а тот способ, который достигал этого значений, получал один балл. Способ, набравший наибольшее количество баллов лучше вычислял значение функции. Приведу сюда таблицу сумм баллов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Функция | Прямая сумма | Попарная сумма | Обратная сумма | Обратная сумма с массивом |
| Синус | 26653 | 16386 | 25583 | 25583 |
| Косинус | 17789 | 14718 | 21753 | 17812 |
| Экспонента | 20678 | 25743 | 10335 | 38513 |
| Логарифм | 16061 | 24078 | 14952 | 45766 |

*Рис. 5. Таблица сумм баллов за наиболее точное вычисление различных функций для различных способов суммирования. Зеленым цветом обозначен наиболее точный для данный функции способ суммирования, голубым – второй по точности.*

По итогам исследования можно заметить, что для синуса наиболее точной оказалась прямая сумма, немного менее точными стали обе обратные суммы. Для косинуса же обратная сумма была заметно точнее прямой суммы и обратной суммы с массивом. Данные между синусом и косинусом так различаются, поскольку вблизи точки 0 косинус находится вблизи 1, что понижает общую точность функций. Попарная сумма оказалась наименее точной, так как для этих функций характерна разность знаков соседних членов, а попарная сумма эффективна только в случае, если соседние члены достаточно разных порядков. В случае с тригонометрическими функциями, сами попарные суммы оказывались разных порядков больше, чем обычные члены в прямой сумме. Прямая сумма оказалась формально точнее обратных для синуса за счёт значений при больших . Для косинуса прямая сумма и обратная сумма с массивом оказались практически равной точности, а так как обратная сумма имеет свойство занижать результат, то при больших она оказалась формально точнее.

Для экспоненты и логарифма ситуация другая: поскольку для обеих функций не характерна разница знаков в соседних членов, но характерна быстрая сходимость, попарная сумма оказалась довольно точной. Обратная сумма с массивом стала наиболее точной как при маленьких , за счет точного вычисления последних битов, так и при больших , за счет точных первых битов и порядка благодаря первому члену суммы, который не суммировался с соседним как в случае с попарной суммой.

# Заключение

По результатам проведенных экспериментов были сделаны следующие выводы:

* Все способы недостаточно точны из-за самого принципа многочлена Маклорена: чем дальше от , тем меньше точность. Из-за этого при достаточно больших некоторые способы оказывались формально точнее, хотя все они были далеки от действительного значения.
* Для расчёта тригонометрических функций довольно точным оказался способ прямого суммирования из-за разницы знаков соседних членов. Для расчёта экспоненты и логарифма точным оказался способ попарного суммирования из-за большой разности между соседними членами. Для всех 4-х функций точным был способ обратного суммирования с массивом. Обычный способ обратного суммирования был менее точным из-за погрешности получения последнего члена.

# Приложение

Код файла sumseries.cpp с реализацией различных типов суммирующих функций:

float nextsin(float x, size\_t i) {

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

}

float nextcos(float x, size\_t i) {

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

}

float nextexp(float x, size\_t i) {

return x / i;

}

float nextlog(float x, size\_t i) {

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

}

float convergesin(float x) { return x;}

float convergecos(float x) { return x;}

float convergeexp(float x) { return x;}

float convergelog(float x) { return ((x - 1) / (x + 1));}

float direct\_sum(float x,

float a0,

float (\*next)(float, size\_t),

float (\*transform)(float),

size\_t n) {

float z = transform(x);

float prev = a0;

float res = prev;

for (size\_t i = 1; i <= n; i++) {

prev \*= next(z, i);

res += prev;

}

return res;

}

float partial\_sum(float x,

float a0,

float (\*next)(float, size\_t),

float (\*transform)(float),

size\_t n) {

float z = transform(x);

float prev = a0;

float prefs = 0;

float res = prev;

for (size\_t i = 1; i <= n; i+=2) {

prefs = prev \* next(z, i) + prev \* next(z, i) \* next(z, i + 1);

prev \*= next(z, i) \* next(z, i + 1);

res += prefs;

}

if (n % 2 != 0) res += prev \* next(z, n);

return res;

}

float backward\_sum(float x,

float a0,

float (\*next)(float, size\_t),

float (\*transform)(float),

size\_t n) {

float z = transform(x);

float res = 0;

float prev = a0;

for (size\_t i = 1; i <= n; i++) prev \*= next(z, i);

res = prev;

if (prev == 0.0) return 0.0;

for (size\_t i = n; i > 0; i--) {

prev /= next(z, i);

res += prev;

}

return res;

}