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

**Отчёт по лабораторной работе**

**«Ряды Тейлора»**

**Выполнила**:

студентка группы 3823Б1ПМ1

Булычева Л. С.

**Проверил**:

преподаватель каф. ВВСП,

Волокитин В.Д.

Нижний Новгород

2023

**Содержание**

[Постановка задачи 3](#__RefHeading___1)

[Метод решения 3](#__RefHeading___2)

[Руководство пользователя 4](#__RefHeading___3)

[Описание программной реализации 4](#__RefHeading___4)

[Заключение 12](#__RefHeading___5)

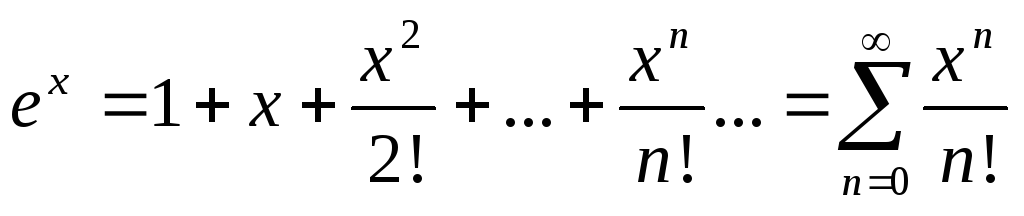
[Список литературы 13](#__RefHeading___6)

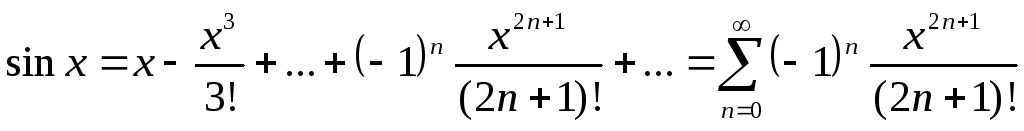
[Приложение 14](#__RefHeading___7)

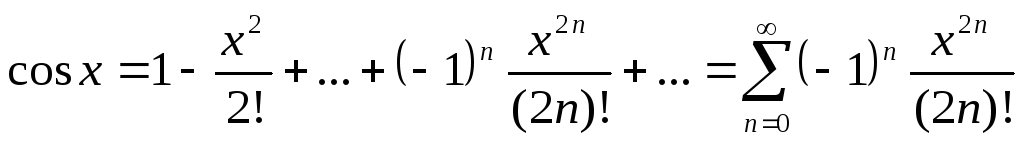
# Постановка задачи

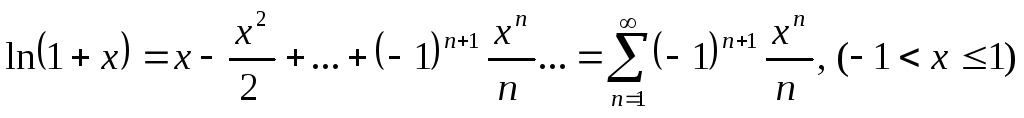
В данной работе необходимо реализовать ряды Маклорена для следующих математических функций: exp(x), sin(x), cos(x), ln(1+x) – на языке С. Нужно посчитать ряды тремя разными видами суммирования: прямым, обратным и попарным. А также выяснить какой способ суммирования наиболее точный.

Ряды Тейлора:









# **Метод решения**

Виды суммирования:

1. Прямая сумма – это вид суммирования от первого члена последовательности к последнему по порядку, то есть последним членом суммы будет последний член ряда. Таким образом происходит суммирование от наибольших членов ряда к маленьким.
2. Обратная сумма – это вид суммирования от последнего члена последовательности к первому по порядку, то есть последним членом суммы будет последний член ряда. Таким образом происходит суммирование от наименьших членов ряда к большим.
3. Попарная сума – это вид суммирования когда члены ряда складываются попарно, то есть 1 член ряда складывается со 2, 3 член ряда складывается с 4 и так далее, в конце складывается n-1 член ряда с n-ым.

Для решения поставленной задачи требуется использовать все три метода суммирования. Для проведения всех трех видов суммирования была разработана формула для вычисления промежуточного члена последовательности (pi). Для избежания переполнения при использовании факториалов, каждый последующий член ряда вычислялся путем умножения предыдущего на промежуточный член.

1. Pi(exp) = x/i
2. Pi(sin) = - x^2 / ( 2 \* i \* ( 2 \* i + 1))
3. Pi(cos) = - x^2 / (( 2\* i \* (2 \* I – 1))
4. 4)Pi(ln) = - x \* i / ( i + 1)

где x – вводимый пользователем аргумент в радианах, i – номер члена ряда.

# **Руководство пользователя**

При запуске программы пользователю дается возможность ввести значение «х» современной арабской записью (цифрами) слева направо, а затем выбрать для какой математической функции программа будет считать сумму ряда Маклорена. На вывод программы будут подаваться значения этой функции вычисленной тремя алгоритмами: прямым, обратным и попарным суммированием.

# **Описание программной реализации**

Программа начинает свою работу с функции main(). Она определяет диапазон членов ряда Маклорена (до 15), а также запрашивает у пользователя ввод значения “x” и математическую функцию(exp, sin, cos, ln), затем она реализует вызов следующих функций для реализования трех видов суммирования и выводит их на консоль для последующей проверки на корректность.

Функции:

float f\_exp1(float x, int i) {

return x / i;

}

float f\_sin1(float x, int i) {

return -(x \* x) / (2 \* i \* (2 \* i + 1.0f));

}

float f\_cos1(float x, int i) {

return -(x \* x) / (2 \* i \* (2 \* i - 1.0f));

}

float f\_log1(float x, int i) {

return -(x \* i) / (i + 1.0f);

}

-функции для нахождения члена ряда различных математических функций(exp, sin, cos, ln).

float sum1(float x0, float x, int n, float(\*f)(float, int)) {

float xi, res;

xi = res = float(x0);

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

xi \*= f(x, i);

res += xi;

}

return res;

}

-функция прямого суммирования с указателем на функцию для нахождения члена ряда (рекурсия в цикле функции - нахождение членов ряда, учитывая предыдущие)

x0 - начальное значение (либо х, либо 1 - в зависимости от выбранной функции)

float ln\_pr(float x, int n, float(\*f)(float, int)) {

float res = 0, xi = x;

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

res += xi;

if (i == 0) xi \*= (-1) \* x / (float)(i + 1);

else xi \*= f(x, i);

}

return res;

}

функция прямого суммирования для логарифма, имеет схожий алгоритм с общей функцией прямого суммирования, но при выполнении условия, что переменная i = 0, не используем ее для нахождения члена ряда в произведении.

float sum2(float x0, float x, int n, float(\*f)(float, int)) {

float xi, res = 0, v;

xi = float(x0);

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

v = f(x, i);

if (v == 0) {

n = i - 1;

break;

}

xi \*= v;

}

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

res += xi;

xi = xi / (f(x, i));

}

return res + float(x0);

}

-функция для нахождения обратной суммы. Отличие состоит в том, что вначале мы находим последний член последовательности (при помощи цикла), а дальше запускаем цикл, в котором с каждым ходом мы получаем предыдущий член посредством выражения xi = xi / (f(x, i));

float sum3(float x0, float x, int n, float(\*f)(float, int)) {

float xi, xn, res;

xi = res = float(x0);

for (int i = 1; i <= n; i = i + 2) {

xi \*= f(x, i);

xn = xi;

res += xi;

if (i + 1 < n) {

xn \*= f(x, i + 1);

xi = xn;

res += xn;

}

}

return res;

}

-функция попарного суммирования, именно поэтому в цикле i = i + 2 (при помощи этого с 1ого элемента мы переходим на 3-ий и т.д.). В цикле мы делаем проверку, чтобы не учесть элемент, который не входит в общее число членов ряда

**Подтверждение корректности**

Для подтверждения корректности на консоль выводятся поочередно сначала значения прямого, обратного и попарного суммирования для выбранной математической функции(exp, sin, cos, ln). Таким образом пользователь имеет возможность сравнить полученные значения при реализации этих алгоритмов с официальными данными для введенного аргумента.

**Результаты экспериментов**

Эксперименты проводились следующим образом: сначала происходило суммирование различными способами (прямое, обратное и попарное), затем все это выводилось на консоль. На последнем шаге происходила проверка на корректность, которая заключалась в том, чтобы сравнить официальные данные синуса, косинуса, экспоненты и натурального логарифма с полученными в ходе эксперимента.

Для наиболее удобного определения корректности результатов были сделаны таблицы значений и разницы показаний, а также графики соответствий.

В таблицах значений написаны значения функций в точке x с различными способами суммирования. В таблицах разницы значений показана разница между официальными данными и данными полученными в ходе работы разными способами суммирования.

Таблица 1 - Значения для exp(x).

|  |  |  |  |
| --- | --- | --- | --- |
| x | -2 | 2 | 3 |
| official | 0.13533528 | 7.38905609 | 20.08553692 |
| прямая | 0.13509698 | 7.38871288 | 20.06339455 |
| обратная | 0.13537920 | 7.38899469 | 20.07966614 |
| попарная | 0.13509698 | 7.38871288 | 20.06339455 |

Таблица 2 - Разница значений exp(x).

|  |  |  |  |
| --- | --- | --- | --- |
| official | 0 | 0 | 0 |
| прямая | 0.0002383 | 0.00034321 | 0.02214237 |
| обратная | 0.00004392 | 0.0000614 | 0.00587078 |
| попарная | 0.0002383 | 0.00034321 | 0.02214237 |

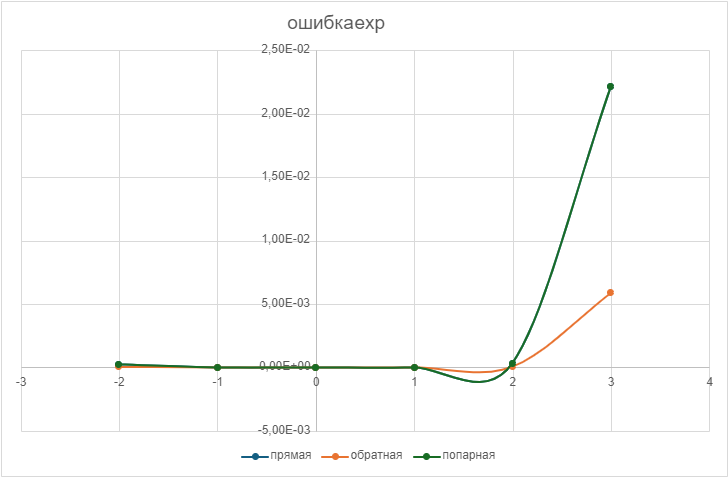


Рис.1. График ошибки exp.

Можно заметить что наиболее точная сумма для exp(x) получилась обратная сумма, прямое и попарное же суммирование дают менее точные значения. При более больших значения начинает уменьшаться точность вычислений.

Таблица 3 – Значения для sin(x).

|  |  |  |  |
| --- | --- | --- | --- |
| x | -2 | 3 | 5 |
| official | -0.90929742 | 0.14112000 | -0.95892427 |
| прямая | -0,90929741 | 0,14111987 | -0,95893365 |
| обратная | -0,90929747 | 0,14112067 | -0,95892334 |
| попарная | -0,90929742 | 0,14111987 | -0,95893366 |

Таблица 4 – Разница значений sin(x).

|  |  |  |  |
| --- | --- | --- | --- |
| official | 0 | 0 | 0 |
| прямая | -0.00000001 | 0.00000013 | 0.00000938 |
| обратная | 0.00000005 | -0.00000067 | -0.00000093 |
| попарная | -0.00000002 | 0.00000013 | 0.00000939 |

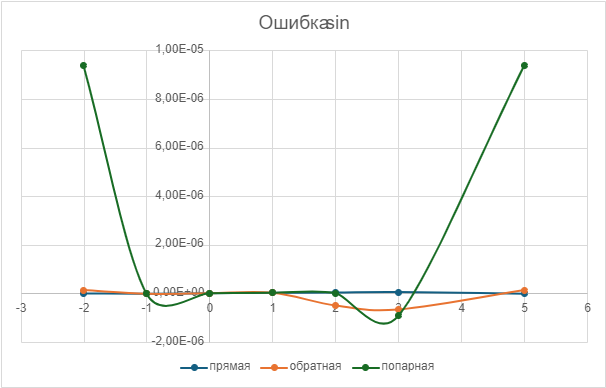


Рис.2. График ошибки sin.

Из таблиц мы видим, что наиболее точным суммированием для функции sin(x) является прямое. Наименее точным же является попарная сумма.

Таблица 5 – Значения для cos(x).

|  |  |  |  |
| --- | --- | --- | --- |
| x | -2 | 3 | 5 |
| official | -0.41614683 | -0,98999249 | 0.28366218 |
| прямая | -0,41614681 | -0,98999250 | 0,28362492 |
| обратная | -0,41614699 | -0,98999262 | 0,28366470 |
| попарная | -0,41614681 | -0,98999250 | 0,28362492 |

Таблица 6 – Разница значений cos(x).

|  |  |  |  |
| --- | --- | --- | --- |
| official | 0 | 0 | 0 |
| прямая | -0.00000002 | 0.00000001 | 0.00003726 |
| обратная | 0.00000016 | 0.00000013 | 0.00003748 |
| попарная | -0.00000002 | 0.00000001 | 0.00003726 |

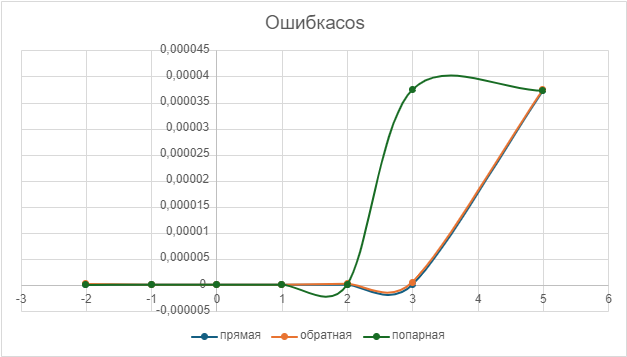


Рис.3. График ошибки cos.

Из таблицы видим, что для функции cos(x) наиболее точными суммами являются прямая и попарная сумма.

Таблица 7 - Значения для ln(x+1).

|  |  |  |  |
| --- | --- | --- | --- |
| x | 0.2 | 0.5 | 0.8 |
| official | 0.18232155 | 0.40546510 | 0.58778666 |
| прямая | 0,18232158 | 0,40553230 | 0,59401292 |
| обратная | 0,18232155 | 0,40547901 | 0,59108448 |
| попарная | 0,18232156 | 0,40543464 | 0,58327550 |

Таблица 8 – Разница значений ln(ч+1).

|  |  |  |  |
| --- | --- | --- | --- |
| official | 0 | 0 | 0 |
| прямая | -0.00000003 | -0.0000672 | -0.00622626 |
| обратная | 0 | -0.00001391 | -0.00329782 |
| попарная | -0.00000001 | 0.00003046 | 0.00451116 |

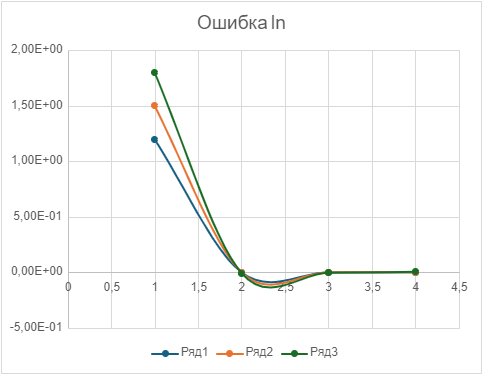


Рис.3. График ошибки ln.

Для функции ln(x+1) наиболее точным суммированием является обратное, наименее же точной оказалось прямая сумма.

.

# **Заключение**

По результатам проведенных экспериментов можно сделать следующие выводы

1. 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.
2. Для тригонометрических функций наиболее корректным алгоритмом суммирования является прямая сумма, изменение в результатах в отличии от обратного и попарного суммирования практически отсутствуют. Для натурального логарифма наиболее точным оказалось обратное суммирование, а наименее точным прямое. При расчете экспоненты более точными были результаты обратного суммирования. Прямое и попарное суммирования практически не отличались.
3. Таким образом можно сделать вывод, что для разный функций наиболее корректны разные суммирования.

# **Список литературы**

1. Демидович Б. П. “Сборник задач и упражнений по математическому анализу. Учебное пособие для вузов”/ Б. П. Демидович. – М.: ООО «Издательство Астрель»: ООО «Издательство АСТ», 2005. (дата обращения – 20.02.2024).

# **Приложение**

float f\_exp1(float x, int i) {

return x / i;

}

float f\_sin1(float x, int i) {

return -(x \* x) / (2 \* i \* (2 \* i + 1.0f));

}

float f\_cos1(float x, int i) {

return -(x \* x) / (2 \* i \* (2 \* i - 1.0f));

}

float f\_log1(float x, int i) {

return -(x \* i) / (i + 1.0f);

}

float sum1(float x0, float x, int n, float(\*f)(float, int)) {

float xi, res;

xi = res = float(x0);

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

xi \*= f(x, i);

res += xi;

}

return res;

}

float ln\_pr(float x, int n, float(\*f)(float, int)) {

float res = 0, xi = x;

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

res += xi;

if (i == 0) xi \*= (-1) \* x / (float)(i + 1);

else xi \*= f(x, i);

}

return res;

}

float sum2(float x0, float x, int n, float(\*f)(float, int)) {

float xi, res = 0, v;

xi = float(x0);

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

v = f(x, i);

if (v == 0) {

n = i - 1;

break;

}

xi \*= v;

}

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

res += xi;

xi = xi / (f(x, i));

}

return res + float(x0);

}

float sum3(float x0, float x, int n, float(\*f)(float, int)) {

float xi, xn, res;

xi = res = float(x0);

for (int i = 1; i <= n; i = i + 2) {

xi \*= f(x, i);

xn = xi;

res += xi;

if (i + 1 < n) {

xn \*= f(x, i + 1);

xi = xn;

res += xn;

}

}

return res;

}