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**

**«Ряды Тейлора»**

**Выполнил**:

студент группы 3822Б1ПМ1

Сергеев М.О.

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2022

**Содержание**

[Постановка задачи 3](#_Toc26962562)

[Метод решения 4](#_Toc26962563)

[Руководство пользователя 5](#_Toc26962564)

[Описание программной реализации 6](#_Toc26962565)

[Подтверждение корректности 7](#_Toc26962566)

[Результаты экспериментов 8](#_Toc26962567)

[Заключение 9](#_Toc26962568)

[Приложение 10](#_Toc26962569)

# Постановка задачи

Цель – изучить, реализовать и сравнить алгоритмы вычисления значений синуса, косинуса, натурального логарифма и экспоненты.

Задачи:

1. Изучить и реализовать алгоритмы.
2. Проверить корректность работы алгоритмов.
3. Сравнить точность алгоритмов.

# Метод решения

Значения функций sin(x), cos(x), exp(x), log(x) вычисляются, как сумма ряда Тейлора. Каждое слагаемое в ряде вычисляется рекурсивно. Суммирование членов ряда выполняется двумя способами: в прямом и обратном порядке.

# Руководство пользователя

На вход программа получает 3 параметра.

1. Целое число от 0 до 4, соответствующее выбору функции.
2. Действительное число – аргумент функции.
3. Натуральное число – количество вычисляемых слагаемых в ряде Тейлора.

Выводит: первое - значение встроенной в стандартную библиотеку функции, второе - значение функции, вычисленное прямым суммированием ряда Тейлора, третье - значение функции, вычисленное обратным суммированием ряда Тейлора; абсолют разности первого и второго значения, абсолют разности первого и третьего значения.

# Описание программной реализации

double mysin(double x, int i) – вычисляет i-ый член ряда Тейлора для синуса.

double mycos(double x, int i) – вычисляет i-ый член ряда Тейлора для косинуса.

double myexp(double x, int i) – вычисляет i-ый член ряда Тейлора для экспоненты.

double mylog(double x, int i) - вычисляет i-ый член ряда Тейлора для логарифма.

double mylog2(double z, int i) - вычисляет i-ый член ряда Тейлора для логарифма.

double sum1(double(\*tmp) (double, int), double x) – суммирует члены ряда Тейлора в прямом порядке.

double sum2(double(\*tmp) (double, int), double x) – суммирует члены ряда Тейлора в обратном порядке.

# Подтверждение корректности

Для подтверждения корректности вычисляются значения встроенных в стандартную библиотеку math.h функций синуса, косинуса, экспоненты, логарифма от тех же аргументов, что и в реализованных в программе функциях.

# Результаты экспериментов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Функция | Аргумент | Значние стандартной функции | Sum1\* | Sum2\*\* | |Sin(x) - Sum1| | |Sin(x) – Sum2| |
| Sin x | 31.4 | -0.0159258626 | -0.0157251714 | -0.0157832742 | 0.0002006912 | 0.0001425884 |
| -29.53 | 0.9507560299 | 0.9508066158 | 0.9507989312 | 0.0000505859 | 0.0000429012 |
| 27.325 | 0.8130278550 | 0.8130339745 | 0.8130329436 | 0.0000061195 | 0.0000050886 |
| Cos x | 31.4 | 0.9998731754 | 1.0001162684 | 1.0000223304 | 0.0002430930 | 0.0001491550 |
| -35.256 | -0.7658199930 | -0.7727133092 | -0.7703500952 | 0.0068933162 | 0.0045301022 |
| 29.19 | -0.6092625002 | -0.6092221528 | -0.6092380922 | 0.0000403474 | 0.0000244081 |
| Exp x | -12.5 | 0.00000372665317207867 | 0.00000372665211828096 | 0.00000372665346048962 | 0.00000000000105379772 | 0.00000000000028841094 |
| -13.31 | 0.00000165783172361451 | 0.00000165782334240362 | 0.00000165783051109258 | 0.00000000000838121089 | 0.00000000000121252192 |
| 8.25 | 3827.62582143990630356711 | 3827.62582143990584881976 | 3827.62582143990630356711 | 0.00000000000045474735 | 0.00000000000000000000 |
| Log x | 9.125 | 2.21101789946855520341 | 2.21101789946855564750 | 2.21101789946855520341 | 0.00000000000000044409 | 0.00000000000000000000 |
| 18.18 | 2.90032208874933283838 | 2.90032208874933017384 | 2.90032208874933328246 | 0.00000000000000266454 | 0.00000000000000044409 |
| 31.35 | 3.44521426707892963748 | 3.44521426707892963748 | 3.44521426707892919339 | 0.00000000000000000000 | 0.00000000000000044409 |

Sum1\* - суммирование в прямом порядке.

Sum2\*\* - суммирование в обратном порядке.

В результате эксперимента видно, что оба метода не всегда дают точный результат, однако видно, что обратный метод суммирования в большинстве случаев точнее.

# Заключение

В результате проделанной работы изучены и реализованы алгоритмы вычисления значений функций синуса, косинуса, логарифма, экспоненты. Подтверждена корректность алгоритмов. Проверена и сопоставлена точность вычисления.

# Приложение

double mysin(double x, int i)

{

double a = x, b = x;

int j=1;

a = b = x;

while (j <= i)

{

b = x \* x \* a / (2 \* j \* (2 \* j + 1));

a = b;

j++;

}

return pow(-1, j+1) \* b;

};

double mycos(double x, int i)

{

double a = x, b = x;

int j = 1;

a = b = 1;

while (j <= i)

{

b = x \* x \* a / ((2 \* j) \* (2 \* j - 1));

a = b;

j++;

}

return pow(-1, j+1) \* b;

};

double myexp(double x, int i)

{

double a = x, b = x;

int j = 1;

a = b = 1;

while (j <= i)

{

b = x \* a / j;

a = b;

j++;

}

return b;

};

double mylog(double x, int i)

{

double a = fabs(x), b = fabs(x);

int j = 1;

if (i == 0) return x;

else

{

while (j <= i)

{

b = fabs(x) \* a \* j / (j + 1);

a = b;

j++;

}

if (x >= 0) return b \* pow(-1, j+1);

else return -b;

}

};

double mylog2(double z, int i)

{

double a = (1-z)/(1+z), b = (1 - z) / (1 + z);

int j = 1;

if (i == 0) return -2\*b;

{

while (j <= i)

{

b = ((1 - z) / (1 + z)) \* ((1 - z) / (1 + z)) \* a \* (2\*j-1) / (2\*j + 1);

a = b;

j++;

}

return -2 \* b;

}

};

double sum1(double(\*tmp) (double, int), double x)

{

double res1=0;

for (int k = 0; k < C; k++)

{

//printf("%.10lf\n", tmp(x, k));

res1 += tmp (x, k) ;

}

return res1;

};

double sum2(double(\*tmp) (double, int), double x)

{

double res1 = 0;

for (int k = C-1; k > - 1; k--)

{

//printf("%.10lf\n", tmp(x, k));

res1 += tmp(x, k);

}

return res1;

};