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 синуса, косинуса, экспоненты и натурального логарифма»**

**Выполнил**:

студент группы 3821Б1ПМ2

Ермолаев Д.А.

**Проверил**:

преподаватель каф. МОСТ,

Волокитин В.Д.

Нижний Новгород

2022

Оглавление

**Постановка задачи3**

**Методы решения4**

**Руководство** **пользователя5**

**Описание программной реализации** **6**

**Подтверждение** **корректности7**

**Результаты экспериментов8**

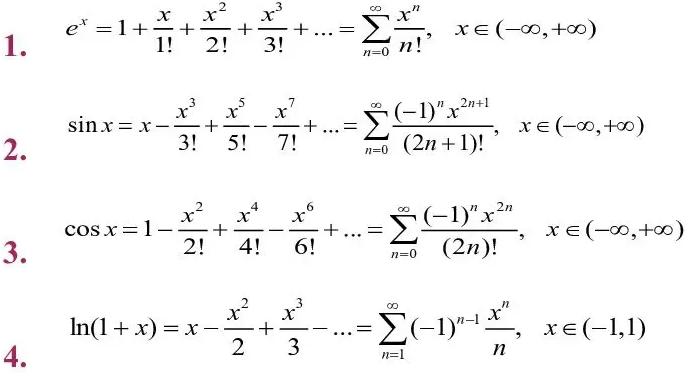
**Заключение12**

**Приложение13**

# Постановка задачи

Целью лабораторной работы является разобрать идею и реализовать на языке программирования Си вычисление функций синуса, косинуса, экспоненты, натурального логарифма с помощью рядов Маклорена. Вычисления я реализовал для данных типа double. Нужно описать программную реализацию и алгоритмы работы данных вычислений. Необходимо подтвердить корректность реализации путём вычисления абсолютной и относительной погрешности данных вычислений. Сделать вывод, какой метод суммирования более точен для определённых функций.

**Методы решения**

Для вычисления дынных функций использовались ряды Маклорена.

Как видно по формулам, ряды Маклорена не имеют определённого последнего элемента, чем больше слагаемых, тем выше точность вычисления. В своей работе я считал функции используя 100 слагаемых. Вычисления функций можно производить 3 способами суммирования:

1. Прямое суммирование (суммирование начинается с первого элемента)
2. Обратное суммирование (суммирование начинается с последнего элемента)
3. Попарное суммирование (сначала суммирование 1-го слагаемого с 3, 5, 7 и т.д и 2-го слагаемого с 4, 6, 8 и т.д, потом сложение этих 2 частей) (имеет смысл для синуса, косинуса и логарифма)

Руководство пользователя

Пользователю предлагается ввести название функции (sin, cos, exp, ln) и число, от которого нужно будет взять эту функцию. Программа высчитывает результат и выводит на экран этот результат, число, от которого берется функция, абсолютную и относительную погрешность от стандартной функции из библиотеки “math.h”.

Описание программной реализации

Библиотеки, которые использовались в работе: “stdio.h”, “math.h”, “string.h”, “stdlib.h”.

Основные функции:

1. double next(char\* namefunc, double pre, int i, double x) – функция возвращает следующее слагаемое ряда Маклорена.
2. double first(char\* namefunc, double x) – функция возвращает первое слагаемое каждой функции, которую нужно вычислить.
3. double directsum(char\* namefunc, double x) – прямое суммирование. Возвращает значение функции от числа x.
4. double reversesum(char\* namefunc, double x) – обратное суммирование. Возвращает значение функции от числа x.
5. double pairwisesum(char\* namefunc, double x) – попарное суммирование. Возвращает значение функции от числа x.
6. void printfunc(char\* namefunc, double func, double x) – выводит на экран число, от которого берётся функция, значение функции от числа x, абсолютную и относительную погрешность.

Подтверждение корректности

Для подтверждения корректности в программе использовал стандартные функции (sin(), cos(), exp(), log()) из стандартной библиотеки “math.h” и высчитывал относительную и абсолютную погрешность относительно результата стандартной функции и результата реализованной функции.

**Результаты экспериментов**

Результаты экспериментов приведены в виде таблицы (1-й столбец - число, от которого берётся функция, 2, 3 и 4 столбцы относительная погрешность (%) разных видов суммирования). Число, от которого берётся функция, менялось от -10 до 10 с интервалом в 1.

**Синус**

|  |  |  |  |
| --- | --- | --- | --- |
| X | Прямое суммирование | Обратное суммирование | Попарное суммирование |
| -10.0 | 0.0000000000140405 | 0.0000000000121222 | 0.0000000000871410 |
| -9.0 | 0.0000000000326506 | 0.0000000000122978 | 0.0000000000566671 |
| -8.0 | 0.0000000000032767 | 0.0000000000000561 | 0.0000000000129834 |
| -7.0 | 0.0000000000003211 | 0.0000000000012167 | 0.0000000000085170 |
| -6.0 | 0.0000000000000397 | 0.0000000000012715 | 0.0000000000089004 |
| -5.0 | 0.0000000000002084 | 0.0000000000002200 | 0.0000000000013314 |
| -4.0 | 0.0000000000000880 | 0.0000000000000880 | 0.0000000000002641 |
| -3.0 | 0.0000000000000393 | 0.0000000000000983 | 0.0000000000013571 |
| -2.0 | 0.0000000000000000 | 0.0000000000000000 | 0.0000000000000000 |
| -1.0 | 0.0000000000000000 | 0.0000000000000000 | 0.0000000000000132 |
| 0.0 | 0.0000000000000000 | 0.0000000000000000 | 0.0000000000000000 |
| 1.0 | 0.0000000000000000 | 0.0000000000000000 | 0.0000000000000132 |
| 2.0 | 0.0000000000000000 | 0.0000000000000000 | 0.0000000000000000 |
| 3.0 | 0.0000000000000393 | 0.0000000000000983 | 0.0000000000013571 |
| 4.0 | 0.0000000000000880 | 0.0000000000000880 | 0.0000000000002641 |
| 5.0 | 0.0000000000002084 | 0.0000000000002200 | 0.0000000000013314 |
| 6.0 | 0.0000000000000397 | 0.0000000000012715 | 0.0000000000089004 |
| 7.0 | 0.0000000000003211 | 0.0000000000012167 | 0.0000000000085170 |
| 8.0 | 0.0000000000032767 | 0.0000000000000561 | 0.0000000000129834 |
| 9.0 | 0.0000000000326506 | 0.0000000000122978 | 0.0000000000566671 |
| 10.0 | 0.0000000000140405 | 0.0000000000121222 | 0.0000000000871410 |

Вывод: из таблицы видно вблизи 0, в среднем, лучше всего считает функция, реализованная прямым суммированием, хоть и разница с обратным суммированием не очень велика. Обратное суммирование лучше всего считает функции, |x| которых сильно превосходит 0. Попарное же суммирование, в основном, проигрывает в обоих случаях.

**Косинус**

|  |  |  |  |
| --- | --- | --- | --- |
| X | Прямое суммирование | Обратное суммирование | Попарное суммирование |
| -10.0 | 0.0000000000066687 | 0.0000000000037975 | 0.0000000001791157 |
| -9.0 | 0.0000000000021080 | 0.0000000000044232 | 0.0000000000205319 |
| -8.0 | 0.0000000000082790 | 0.0000000000020602 | 0.0000000001583307 |
| -7.0 | 0.0000000000008983 | 0.0000000000003387 | 0.0000000000003387 |
| -6.0 | 0.0000000000002313 | 0.0000000000001272 | 0.0000000000001272 |
| -5.0 | 0.0000000000005675 | 0.0000000000004892 | 0.0000000000007632 |
| -4.0 | 0.0000000000000000 | 0.0000000000000170 | 0.0000000000000170 |
| -3.0 | 0.0000000000000112 | 0.0000000000000224 | 0.0000000000000224 |
| -2.0 | 0.0000000000000133 | 0.0000000000000400 | 0.0000000000001467 |
| -1.0 | 0.0000000000000205 | 0.0000000000000000 | 0.0000000000000000 |
| 0.0 | 0.0000000000000000 | 0.0000000000000000 | 0.0000000000000000 |
| 1.0 | 0.0000000000000205 | 0.0000000000000000 | 0.0000000000000000 |
| 2.0 | 0.0000000000000133 | 0.0000000000000400 | 0.0000000000001467 |
| 3.0 | 0.0000000000000112 | 0.0000000000000224 | 0.0000000000000224 |
| 4.0 | 0.0000000000000000 | 0.0000000000000170 | 0.0000000000000170 |
| 5.0 | 0.0000000000005675 | 0.0000000000004892 | 0.0000000000007632 |
| 6.0 | 0.0000000000002313 | 0.0000000000001272 | 0.0000000000001272 |
| 7.0 | 0.0000000000008983 | 0.0000000000003387 | 0.0000000000003387 |
| 8.0 | 0.0000000000082790 | 0.0000000000020602 | 0.0000000001583307 |
| 9.0 | 0.0000000000021080 | 0.0000000000044232 | 0.0000000000205319 |
| 10.0 | 0.0000000000066687 | 0.0000000000037975 | 0.0000000001791157 |

Вывод: ситуация аналогична с синусом – прямое суммирование хорошо считает близь 0, обратное суммирование считает лучше, если |x| сильно превосходит 0, попарное суммирование, в среднем, хуже считает и то, и другое.

**Экспонента**

|  |  |  |
| --- | --- | --- |
| X | Прямое суммирование | Обратное суммирование |
| -10.0 | 0.0000002028299278 | 0.0000002234992354 |
| -9.0 | 0.0000001520350629 | 0.0000001012320587 |
| -8.0 | 0.0000000133483873 | 0.0000000128727234 |
| -7.0 | 0.0000000005222441 | 0.0000000005160971 |
| -6.0 | 0.0000000001496777 | 0.0000000001933301 |
| -5.0 | 0.0000000000123064 | 0.0000000000114053 |
| -4.0 | 0.0000000000012313 | 0.0000000000002273 |
| -3.0 | 0.0000000000001672 | 0.0000000000002369 |
| -2.0 | 0.0000000000000205 | 0.0000000000000820 |
| -1.0 | 0.0000000000000302 | 0.0000000000000000 |
| 0.0 | 0.0000000000000000 | 0.0000000000000000 |
| 1.0 | 0.0000000000000163 | 0.0000000000000000 |
| 2.0 | 0.0000000000000240 | 0.0000000000000000 |
| 3.0 | 0.0000000000000354 | 0.0000000000000000 |
| 4.0 | 0.0000000000000521 | 0.0000000000000000 |
| 5.0 | 0.0000000000000000 | 0.0000000000000192 |
| 6.0 | 0.0000000000000000 | 0.0000000000000000 |
| 7.0 | 0.0000000000000622 | 0.0000000000000207 |
| 8.0 | 0.0000000000000610 | 0.0000000000000153 |
| 9.0 | 0.0000000000000224 | 0.0000000000000112 |
| 10.0 | 0.0000000000000165 | 0.0000000000000495 |

Вывод: ситуация схожа с 2 предыдущими, но разница уже менее заметна. В среднем, обратное суммирование лучше считает функцию и при x вблизи 0, и при |x|, сильно превосходящем 0.

**Натуральный логарифм**

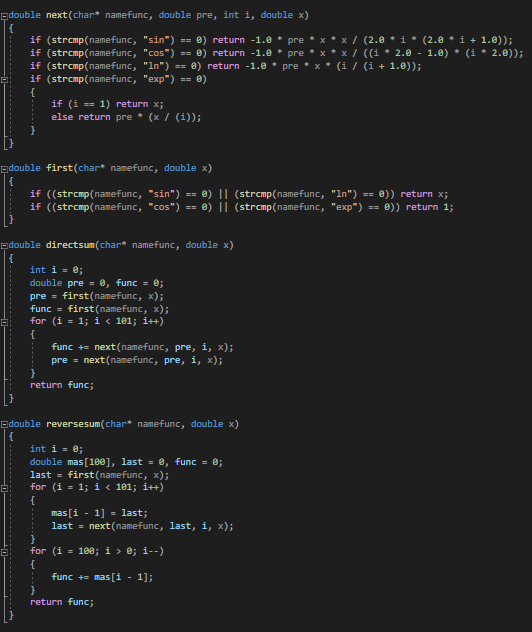
x должен быть в диапазоне (-1;1), для выполнения условий ряда Маклорена. Число, от которого берётся функция, меняется от -0,99 до 0,91 с интервалом в 0,1.

|  |  |  |  |
| --- | --- | --- | --- |
| X | Прямое суммирование | Обратное суммирование | Попарное суммирование |
| -0.99 | 4.6063352193192015 | 4.6842442132563225 | 4.6842442132563225 |
| -0.89 | 0.0000258883500210 | 0.0000293574185161 | 0.0000293574184960 |
| -0.79 | 0.0000000001044315 | 0.0000000001334276 | 0.0000000001334418 |
| -0.69 | 0.0000000000000379 | 0.0000000000000000 | 0.0000000000000190 |
| -0.59 | 0.0000000000000125 | 0.0000000000000000 | 0.0000000000000249 |
| -0.49 | 0.0000000000000330 | 0.0000000000000165 | 0.0000000000000000 |
| -0.39 | 0.0000000000000225 | 0.0000000000000000 | 0.0000000000000225 |
| -0.29 | 0.0000000000000162 | 0.0000000000000000 | 0.0000000000000162 |
| -0.19 | 0.0000000000000132 | 0.0000000000000132 | 0.0000000000000132 |
| -0.09 | 0.0000000000000736 | 0.0000000000000589 | 0.0000000000000736 |
| 0.01 | 0.0000000000008368 | 0.0000000000008368 | 0.0000000000008368 |
| 0.11 | 0.0000000000000133 | 0.0000000000000000 | 0.0000000000000000 |
| 0.21 | 0.0000000000000874 | 0.0000000000000437 | 0.0000000000000728 |
| 0.31 | 0.0000000000000411 | 0.0000000000000206 | 0.0000000000000206 |
| 0.41 | 0.0000000000000323 | 0.0000000000000000 | 0.0000000000000323 |
| 0.51 | 0.0000000000000269 | 0.0000000000000269 | 0.0000000000000269 |
| 0.61 | 0.0000000000000350 | 0.0000000000000000 | 0.0000000000000233 |
| 0.71 | 0.0000000000000621 | 0.0000000000000000 | 0.0000000000000414 |
| 0.81 | 0.0000000004243843 | 0.0000000005291892 | 0.0000000005291331 |
| 0.91 | 0.0000529221806016 | 0.0000587347474243 | 0.0000587347473729 |

Вывод: натуральный логарифм начинает сильно расходится в окрестностях -1 и 1 и появляются большие погрешности при любых видах суммирования. Прямое и попарное суммирования, в среднем, считают приблизительно равно. Обратное же суммирование, в основном, считает лучше при любом x.

**Заключение**

В ходе лабораторной работы на языке программирования Си были реализованы 3 метода суммирования рядов Маклорена для получения наиболее точных результатов вычисления функций синуса, косинуса, экспоненты и натурального логарифма. Были описаны алгоритмы работы данных вычислений, их программная реализация и проведены эксперименты для подтверждения их корректности, методом вычисления погрешностей относительно функций из стандартных библиотек. Также, были определены какие виды суммирования лучше справляются в тех или иных случаях.

**Приложение**

