**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский Авиационный Институт»**

**(Национальный Исследовательский Университет)**

**Институт №8 «Информационные технологии и прикладная математика» Кафедра 806 «Компьютерные науки и прикладная математика»**

Курсовая работа по курсу «Вычислительные системы»

1 семестр

Задание 3

**Автор работы:**

студент 1 курса, гр. М8О-108Б-22 Цирулев Н.В. **Руководитель проекта: Дата сдачи:**

**Оценка**:

Москва, 2022

**ОГЛАВЛЕНИЕ**

[ЗАДАЧА 3](#_Toc124171560)

[ОБЩИЙ МЕТОД РЕШЕНИЯ 4](#_Toc124171561)

[ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ 5](#_Toc124171562)

[ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ 6](#_Toc124171563)

[ОПИСАНИЕ ПРОГРАММЫ 7](#_Toc124171564)

[ПРОТОКОЛ 10](#_Toc124171565)

[ВЫВОД 13](#_Toc124171566)

[ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ 14](#_Toc124171567)

# Задача

Составить программу на языке программирования Си, которая печатает таблицу значений элементарной функции, вычисленной двумя способами: по формуле Тейлора и с помощью функций из стандартной библиотеки языка Си. В качестве аргументов таблицы взять точки разбиения [a,b] на n равных частей (n+1 точка включая концы отрезка), находящихся в рекомендованной области достаточной точности формулы Тейлора. Вычисления по формуле Тейлора проводить по экономной в сложностном смысле схеме с точностью ε\*k, где ε – машинное эпсилон аппаратно реализованного типа для данной ЭВМ, а k – экспериментально подбираемый коэффициент, обеспечивающий приемлемую сходимость. Число итераций должно ограничиваться сверху числом порядка 100. Программа должна сама определять машинное ε и обеспечивать корректные размеры генерируемой таблицы.

22 вариант задания:

Отрезок - [0.0, 1.0]

| Функция | Разложение в ряд |
| --- | --- |
|  |  |

За количество x-ов на отрезке [0.0, 1.0] взято число 15.

# Общий метод решения

Общий метод решения заключается в нахождении значения функции в некоторой точке при помощи двух способов.

Первый способ заключается в использовании функций, встроенных в стандартную математическую библиотеку языка Си «math.c». В стандартной библиотеке имеется функция «expl», которая считает e^x, где x — единственный аргумент функции.

Функция, реализующая вычисление с помощью ряда Тейлора для функции f(x) в окрестности точки a выглядит следующим образом:

Основополагающей вещью в вычислении данной функции является наличие, так называемого, машинного эпсилон, которое является критерием точности вычислений на заданной ЭВМ.

Машинное эпсилон — минимальное число, выразимое на конечной вычислительной машине.

Его можно найти путём сравнения «1 + ε» с «1». Последнее число, при стремлении к нулю, при котором данное выражение выдаст false и будет машинным эпсилон.

Я буду вычислять на каждом шаге итерации n-ое слагаемое ряда Тейлора и, в случае, если данное слагаемое будет меньше k\*ε (где k — эмперически подобранный коэффицент), то далее вычислять ряд Тейлора является бессмысленным, т.к. члены ряда дошли до максимальной точности компьютера.

# Общие сведения о программе

Язык и система программирования: GNU C

Местонахождение файлов \\wsl.localhost\Ubuntu\home\hackerman\

Способ вызова и загрузки:

gcc cw.c -Wall -std=c18 -pedantic -lm

./a.out

# Функциональное назначение

Программа предназначена для выполнения вещественных вычислений значений трансцедентных функций в алгебраической форме с использованием ряда Тейлора.

Ряд Тейлора – это разложение функции в бесконечную сумму степенных

функций. Если функция f(x) имеет непрерывные производные до (n + 1) порядка, то ее можно разложить по формуле Тейлора.

Ранее данный метод использовался для аппаратного вычисления подобных

функций, так как в то время компьютеры были способны только на сложение,

вычитание и умножение. На сегодняшний день аппаратное обеспечение позволяет вычислять трансцендентные функции другими способами, которые более эффективны во всех смыслах.

# Описание программы

Программа работы:

– Определяем стандартные функции языка С, подключая заголовки «math.h» и «stdio.h».

– Определяем функцию вычисления машинного эпсилон.

– Определяем функцию для вычисления члена ряда Тейлора.

– Определяем функцию для вычисления функции при помощи встроенных функций.

– Вычисляем машинное эпсилон и выводим.

– Печатаем таблицу аргументов функций, значений, полученных средствами языка С и ряда Тейлора, количество итераций, запрошенное машиной для вычисления значения функции.

– Конец.

Составленная программа, решающая данную задачу, состоит из 5-ти функций и 5-ти объявленных констант, которые можно менять в тексте программы для изменения точности значений, количества шагов и размеры отрезка [a, b].

|  |  |  |
| --- | --- | --- |
| Название функции | Входные аргументы | Описание функции |
| compute\_epsilon | - | Функция считает машинный epsilon, методом, описанным выше, а именно сравнивая 1+ε и 1. Пока выражение 1 < 1 + ε возвращает true, функция делит epsilon пополам. |
| inner\_func | long double x | Функция вычисляет функцию, данную в задаче при помощи встроенных в язык программирования С средств. Используется функция expl, которая вычисляет экспоненту для long double типа. |
| factorial | long long n | Функция вычисляет факториал числа n, данное во входных аргументах, путём итерирования от 2 до n включительно и умножения ans на i, где ans — ответ, а i — число, которое пробегается от 2 до n. |

Таблица 1. Описание функций программы:

| long double k | Эмпирический коэффицент для eps |
| --- | --- |
| long double eps | Машинный эпсилон |
| long double a,b | Границы отрезка |
| int n | Кол-во итераций |
| int steps | Кол-во отрезков |
| int max\_iters | Максимальное кол-во итераций |
| long double cur\_member | I-ое слагаемое ряда |
| long double sum | Сумма ряда |

Таблица 2. Описание переменных и констант

# Протокол

hackerman@WARMACHINE\_mini:~$ cat cw.c

#include <math.h>

#include <stdio.h>

typedef long double ld;

ld compute\_epsilon(){

ld eps = 1;

while (1 < 1 + eps)

eps /= 2;

return eps;

}

ld inner\_func(ld x){

return (1 + x) \* expl(-x);

}

int factorial(long long n){

ld ans = 1;

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

ans \*= i;

}

return ans;

}

ld teilor\_member(ld x, int n){

ld v = 2 \* ((-1) \* ((n + 1) & 1)) + 1;

v \*= (n-1);

v /= (ld)factorial(n);

v \*= powl(x, n);

return v;

}

int main() {

const ld k = 10e2;

const ld a = 0;

const ld b = 1;

const int steps = 15;

const int max\_iters = 100;

ld step = (b-a) / steps;

ld eps = compute\_epsilon();

printf("Machine epsilon for long double for this system is %.20Lf\n", eps);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("|x | Sum |(1 + x) \* e ^ (-x) | n|\n");

printf("|\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_|\n");

for (ld x = a; x < b + step; x += step) {

int n = 0;

ld cur\_member = 1;

ld sum = 0;

while ((fabsl(cur\_member) > eps \* k && n < max\_iters) || n == 2) {

cur\_member = teilor\_member(x, n);

sum += cur\_member;

n++;

}

printf("|%.2Lf|%.19Lf|%.19Lf|%3d|\n", x, sum, inner\_func(x), n);

}

printf("|\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_|\n");

}

hackerman@WARMACHINE\_mini:~$ gcc cw.c -Wall -std=c18 -pedantic -lm

hackerman@WARMACHINE\_mini:~$ ./a.out

Machine epsilon for long double for this system is 0.00000000000000000005

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|x | Sum |(1 + x) \* e ^ (-x) | n|

|\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_|

|0.00|1.0000000000000000000|1.0000000000000000000| 3|

|0.07|0.9978741173670589202|0.9978741173670589203| 11|

|0.13|0.9918630949153404478|0.9918630949153404478| 13|

|0.20|0.9824769036935782244|0.9824769036935782304| 14|

|0.27|0.9701758952618881261|0.9701758952618883441| 16|

|0.33|0.9553750807650485869|0.9553750807650523339| 19|

|0.40|0.9384480644498563176|0.9384480644498950211| 22|

|0.47|0.9197306584002051000|0.9197306584004823214| 27|

|0.53|0.8995242032471930592|0.8995242032487153936| 32|

|0.60|0.8780986177436185365|0.8780986177504422922| 40|

|0.67|0.8556951983615859334|0.8556951983876533782| 50|

|0.73|0.8325291884681413595|0.8325291885556522441| 66|

|0.80|0.8087921351469114567|0.8087921354109988645| 93|

|0.87|0.7846540503540443737|0.7846540510828729228|100|

|0.93|0.7602653917912884671|0.7602653936792899699|100|

|1.00|0.7357588549435166975|0.7357588823428846432|100|

|\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_|

hackerman@WARMACHINE\_mini:~$

# вывод

В процессе выполнения этого задания, я получил навыки вычисления и дальнейшего использования машинного эпсилон. После генерации таблицы значений заданной функции можно увидеть, что значения совпадают до 10-14 знака после запятой. Из-за того, что существует понятие ограниченности разрядной сетки, вещественные числа имеют диапазон представления в памяти компьютера, что неизбежно приводит к тому, что в вычислениях в окрестности границ этого диапазона возникают погрешности.

На данный момент использование ряда Тейлора для вычисления трансцендентных функций является не оправданным, т. к. они требуют намного больше ресурсов, чем современные методы и имеют меньшую точность.

# Использованые источники

1. Численные методы. Линейная алгебра и нелинейные уравнения. Учебное пособие. — Directmedia, 2014-05-20. — 432 с.
2. Ильин В. А., Садовничий В. А., Сендов Б. Х. Математический анализ, ч. 1, изд. 3, ред. А. Н. Тихонов. М.: Проспект, 2004.
3. Романов Е. Си/Си++. От дилетанта до профессионала.