МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

“Московский Авиационный Институт

(Национальный Исследовательский Университет)”

Факультет информационных технологий и прикладной математики

Кафедра 806 “Вычислительная математика и программирование”

Курсовая работа

по курсу “Вычислительные системы”

1 семестр

Задание 3. Вещественный тип. Приближенные вычисления. Табулирование функций.

Студент: Цирулев Н.В.

Группа: М8О-108Б-22,

№ по списку 22

Руководитель: Сахарин Н.А.

Дата: 09.01.23

Оценка:

Москва, 2023

ОГЛАВЛЕНИЕ

ЗАДАЧА 3
ОБЩИЙ МЕТОД РЕШЕНИЯ 4
ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ... 5
ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ... 6
ОПИСАНИЕ ПРОГРАММЫ 7
ПРОТОКОЛ 9
ВЫВОД.. 12
ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ 13

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]

Таблица 1

Функция	Разложение в ряд
$(1+x)\cdot e^{-x}$	$\frac{(-1)^{(n-1)}\cdot(n-1)}{n!}\cdot x^n$

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:

$$f(x) = f(a) + \frac{f'(a)}{1!} \cdot (x-a) + \frac{f''(a)}{2!} \cdot (x-a)^2 + \ldots + \frac{f^{(n)}(a)}{n!} \cdot (x-a)^n + \ldots$$

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 ε, 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 C, 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 C 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].

Таблица 2: Описание функций программы

Название функции	Входные аргументы	Описание функции
compute_epsilon	-	Функция считает машинный epsilon, методом, описанным выше, а именно сравнивая 1+ε и 1. Пока выражение $1 < 1 + \varepsilon$ 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.
taylor	long double x	Функция считает значение функции по вводимому x
tabulation	const ld a, const ld b, const uint steps	Функция считает значение для каждого x из заданного отрезка

Таблица 3: Описание переменных и констант

int sign	Переменная, отвечающая за знак
Long long factorial	Факториал числа
long double temp	Промежуточная переменная для вычислений
long double eps	Машинный эпсилон
long double a,b	Границы отрезка
int n	Кол-во итераций
int steps	Кол-во отрезков
MAX_ITER 100	Максимальное кол-во итераций
long double cur_member	I-ое слагаемое ряда
long double result	Сумма ряда

ПРОТОКОЛ

```
hackerman@WARMACHINE_mini:~$ cat cw.c
#include <math.h>
#include <stdio.h>

#define MAX_ITER 100

typedef long double ld;
typedef unsigned uint;

ld compute_epsilon() {
ld eps = 1;
while (1 < 1 + eps)
eps /= 2;
return eps;
}

ld inner_func(ld x) {
return (1 + x) * expl(-x);
}

ld taylor(ld x) {

ld eps = compute_epsilon();
int n = 0;
int sign = 1;
ld cur_member = 1;
ld result = 0;
long long factorial = 1;
ld temp = 1;
while ((fabsl(cur_member) > sqrt(eps) && n < MAX_ITER) || n == 2) {
sign *= -1;
cur_member = sign * (n - 1) / (ld)factorial * temp;
```

```
factorial *= (n + 1);
temp *= x;
result += cur_member;
n++;
}
return result;
}

void tabulation(const ld a, const ld b, const uint steps) {
const ld step = (b - a) / steps;
for (ld x = a; x < b + step; x += step) {
printf("|%.3Lf|%.19Lf|%.19Lf|\n", x, inner_func(x), taylor(x));
}
}

int main() {
const ld a = 0;
const ld b = 1;
const uint steps = 15;
printf("_____________________________________________________\n");
printf("| x | Inner function | Sum of row |\n");
printf("|_____|______________________|______________________|\n");
tabulation(a, b, steps);
printf("|_____|______________________|______________________|\n");
}
```

```
hackerman@WARMACHINE_mini:~$ gcc cw.c -Wall -std=c18 -pedantic -lm
hackerman@WARMACHINE_mini:~$ ./a.out
_____________________________________________________
|  x  |   Inner function    |     Sum of row      |
|_____|______________________|______________________|
|0.000|1.0000000000000000000|1.0000000000000000000|
|0.067|0.9978741173670589203|0.9978741173671260913|
```

0.133	0.99186309491534044 78	0.991863094915051 1802
0.200	0.9824769036935782304	0.9824769036938271605
0.267	0.9701758952618883441	0.9701758952662796686
0.333	0.9553750807650523339	0.9553750807636801683
0.400	0.9384480644498950211	0.9384480644502677570
0.467	0.9197306584004823214	0.9197306584028395720
0.533	0.8995242032487153936	0.8995242032481926710
0.600	0.8780986177504422922	0.8780986177480374027
0.667	0.8556951983876533782	0.8556951983782389749
0.733	0.8325291885556522441	0.8325291885574947002
0.800	0.8087921354109988645	0.8087921354171997454
0.867	0.7846540510828729228	0.7846540510816903548
0.933	0.7602653936792899699	0.7602653936757110237
1.000	0.7357588823428846432	0.7357588823328530670
_____	______________________	______________________

hackerman@WARMACHINE_mini:~$

ВЫВОД

В процессе выполнения этого задания, были изучены 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.