МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский Авиационный Институт»
(Национальный Исследовательский Университет)**

Институт№8: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Курсовой проект
по курсу фундаментальная информатика 1 семестра
Задание 3. Вещественный тип. Приближенные вычисления. Табулирование функций

Студент: Калюжный М.С.

Группа: М8О-108Б-22

Преподаватель: Сахарин Н.А.

Подпись:

Оценка:

Москва 2022

СОДЕРЖАНИЕ

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.

11 вариант задания:
Отрезок - [0.1, 0.6]

Функция: (1 - (x^2 / 2)) * cos(x) - (x / 2) * sin(x)
Разложение в ряд: 1 – (3 / 2) * x^2 + … + (-1)^n * (2n^2 + 1) / (2n)! * x^2n

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, имеющихся в стандартной библиотеке «math.h» языка Си.

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

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 (1 + ε = 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-ное слагаемое ряда Тейлора и, в случае если данное слагаемое будет меньше k*ε (где k — экспериментально подобранный коэффициент), то далее вычислять ряд Тейлора является бессмысленным, т.к. члены ряда дошли до максимальной точности компьютера.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

ОС семейства: UNIX

Наименование: Ubuntu

Версия: Ubuntu 22.04 LTS

Интерпретатор команд: bash

Версия: 5.1-6ubuntu1

Компилятор: gcc

ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Программа предназначена для выполнения вещественных вычислений значений трансцендентных 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. Но на сегодняшний день аппаратное обеспечение позволяет вычислять трансцендентные функции другими способами, которые более эффективны во всех смыслах.

ОПИСАНИЕ ПРОГРАММЫ

Программа работы:

– Подключаем 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

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 C средств. Используется функция powl, которая вычисляет степень числа 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

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

```
#include <math.h>
#include <stdio.h>


long double compute_epsilon();
long double inner_func(long double x);
int factorial(long long n);
long double teilor_series(long double x, int n);
void printtab(long double k, long double a, long double b, int steps, int max_iters);


int main() {
   long double k = 10e-40;
   long double a = 0.1l;
   long double b = 0.6l;
   int steps = 15;
   int max_iters = 100;
   printtab( k, a, b, steps, max_iters);
}


long double compute_epsilon(){
   long double eps = 1;
   while(1 < 1 + eps)
      eps /= 2;
   return eps;
}
```

```
long double inner_func(long double x){
    return (1 - powl(x,2) / 2) * cos(x) - x/2 * sin(x);
}

int factorial(long long n){
    long double ans = 1;
    for (long long i = 2; i <= n; ++i) {
        ans *= i;
    }
    return ans;
}

long double teilor_series(long double x, int n){
    long double v = pow(-1, n);
    v *= (2 * n * n + 1);
    v /= 2 * (long double)factorial(n);
    v *= powl(x, 2 * n);
    return v;
}

void printtab(long double k, long double a, long double b, int steps, int max_iters){
    long double step = (b-a)/steps;
    long double eps = compute_epsilon();
    printf("Machine epsilon for long double for this system is %.20Lf\n", eps);

printf("_____
_____\n");
    printf("|x   | Sum                | (1 - (x^2 / 2)) * cos(x) - (x / 2) * sin(x) |  n|\n");
```

```
printf("|____|_____________________|_____________________________________
_____________|___|\n");

        for(long double x = a; x < b + step; x += step){
            int n = 0;
            long double cur_member = 1;
            long double sum = 0;
            while((fabsl(cur_member) > eps * k && n < max_iters) || n == 2){
                cur_member = teilor_series(x, n);
                sum += cur_member;
                n++;
            }
            printf("|%.2Lf|%.19Lf|%.43Lf|%3d|\n", x, sum, inner_func(x), n);
        }

printf("|____|_____________________|_____________________________________
_____________|___|\n");
    }
```

```
mimik@mimik-VirtualBox:~$ gcc kp3.c -lm
mimik@mimik-VirtualBox:~$ ./a.out
    Machine epsilon for long double for this system is 0.0000000000000000005
    _____________________________________________________________________
    |x   | Sum                | (1 - (x^2 / 2)) * cos(x) - (x / 2) * sin(x) |  n|
    |____|____________________|_____________________________________________|___|
    |0.10|0.49502491687458402 68|0.98503747361929428372640665800830106491048 35| 26|
    |0.13|0.49118965730908880 07|0.97345170366349909957725747289281059693166 65| 29|
    |0.17|0.48630223855817419 60|0.95862212006676650473922796091841291854507 29| 33|
    |0.20|0.48039471957616160 47|0.94059831320491067208934554688681828338303 60| 36|
```

```
|0.23|0.4735055595626277271|0.919440655845231895944725392100238536841061 4| 40|
|0.27|0.465679201055675984 4|0.895220161247672487883544206743735571762954 4| 44|
|0.30|0.4569655926356140933|0.868018316115752779874460004494451936807308 8| 48|
|0.33|0.4474196584071848872|0.837926888720115793467418247031375244660012 0| 52|
|0.37|0.4371007221372124583|0.805047712565679633831168537927425177258555 8| 57|
|0.40|0.4260718944831056692|0.769492446020924190138844805941786830771889 1| 63|
|0.43|0.4143994321477129546|0.731382308374646569379927407705821451600058 9| 68|
|0.47|0.4021520780327861106|0.690847792831533718822378115076432436580944 3| 75|
|0.50|0.3894003915357024380|0.648028357003025412552688067080453038215637 2| 82|
|0.53|0.3762160780446516245|0.603072091494112516596188405815226474260271 1| 91|
|0.57|0.3626713264422267435|0.556135367229855207980761949704273661154729 8|100|
|0.60|0.3488381630355159541|0.507382462207425617380963933644721919336007 0|100|
|____|_____|_____|___|
mimik@mimik-VirtualBox:~$
```

ВЫВОД

В процессе выполнения данного курсового проекта были получены навыки вычисления и дальнейшего использования так называемого «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.
ermak.cs.nstu.ru. Проверено 25 мая 2015.