

Московский авиационный ИНСТИТУТ

(национальный исследовательский университет)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

1 семестр

Курсовой проект

По курсу «Вычислительные системы»

Задание III

Выполнил: Ляпин И. А.

Группа: М8О-104Б-22

Руководитель: Потенко М. А.

Оценка: _____

Дата: _____

Москва, 2022

Содержание

Цель работы	2
Ход работы	2
Алгоритм	4
Список переменных	5
Список функций	5
Общие сведения о программе	5
Код программы	6
Результат работы программы	7
Вывод	8
Дополнительное задание	9
Справочник	9

Цель работы:

В данной лабораторной работе необходимо составить программу на языке Си, для вычисления некоторой функции $f(x)$ двумя способами: обычным методом и с помощью вычисления по ряду Тейлора. Аргументами функции являются все точки, принадлежащие отрезку $[a, b]$, причем данный отрезок необходимо разбить на n равных частей. Необходимо чтобы все под-отрезки отрезка $[a, b]$ соответствовали некоторой заданной области точности. Значение подобранной точности должно быть численно равно произведению значений машинного эпсилона и значения k , где k - некоторый подобранный коэффициент точности, подобрав который, можно обеспечить необходимую нам сходимость. Стоит учесть, что число итераций при подсчете формулы Тейлора не должно превышать - 100. Результаты необходимо вывести в виде таблицы, состоящей из значений: аргумента функций, суммы ряда Тейлора, функции обычным методом, число затраченных итераций по Тейлору.

Ход работы

В первую очередь стоит сказать, что такое ряд Тейлора:

Ряд Тейлора - разложение функции в бесконечную сумму степенных функций.

Частный случай разложения в ряд Тейлора в нулевой точке называется рядом Маклорена.

Благодаря разложению некоторой функции $f(x)$ на сумму членов ряда, можно «повторить» саму функцию. То есть при увеличении членов ряда Тейлора значение суммы ряда будет равен значению раскладываемой функции с некоторой погрешностью.

Продemonстрируем это наглядно на графике нашей функции:

8	$-\frac{1}{5} - \frac{2x}{5^2} - \frac{4x^2}{5^3} - \dots - \frac{2^{n-1}x^{n-1}}{5^n}$	0.0	2.0	$\frac{1}{2x-5}$
---	---	-----	-----	------------------

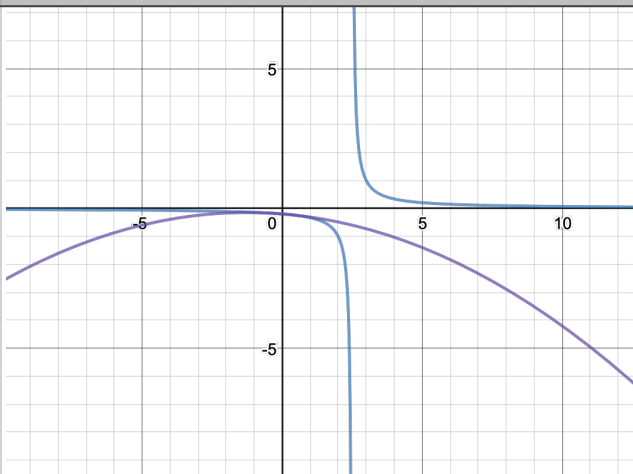
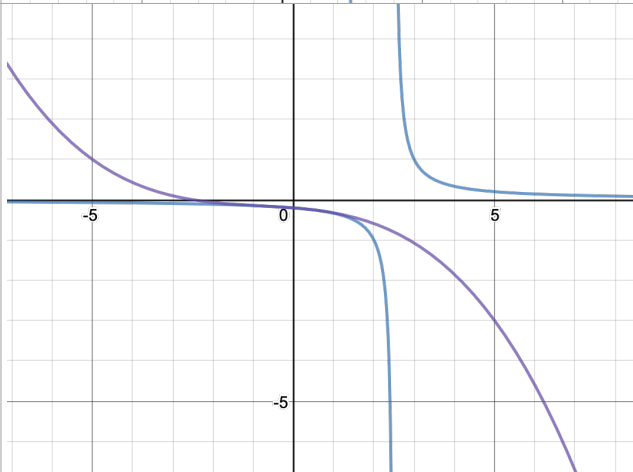
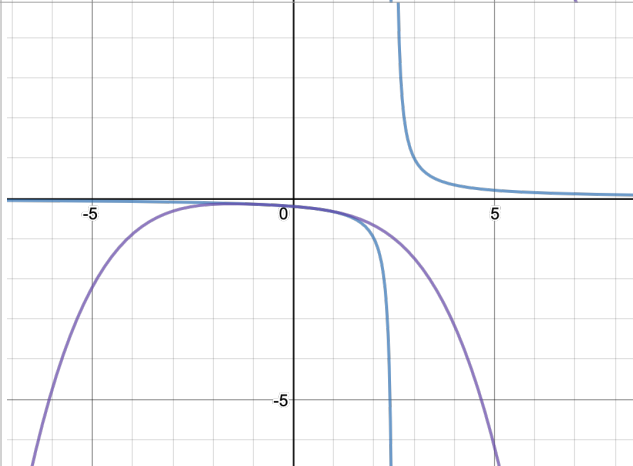
Подсчитаем значения суммы ряда 5-ти членов и сравним её со значением $f(x)$

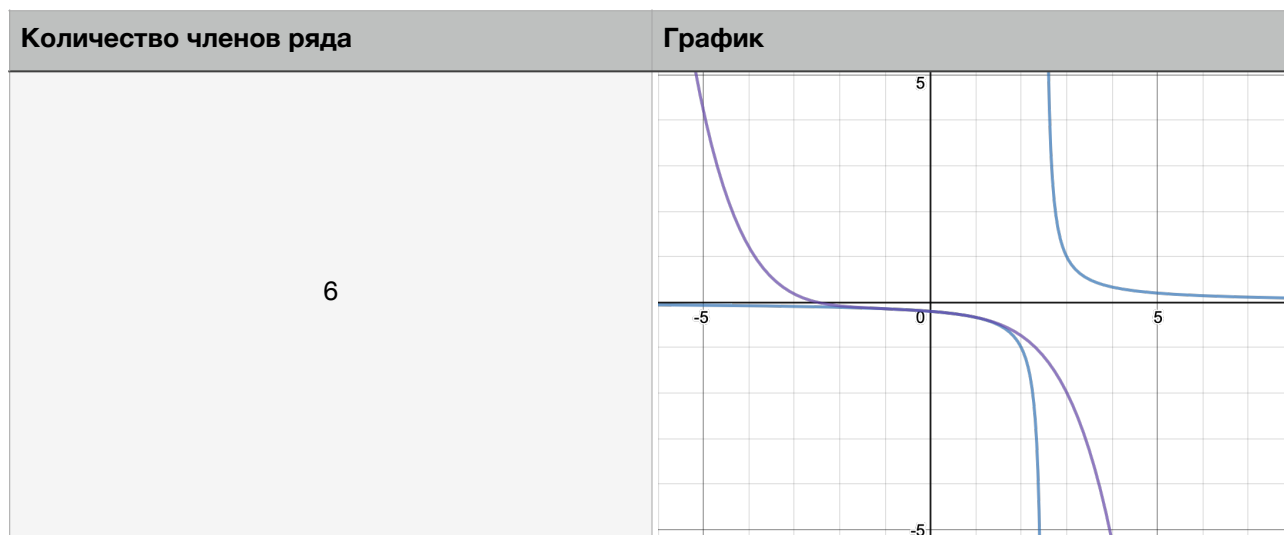
Возьмем $x = 0$ для удобства.

Пусть синим цветом изображена функция

$$\begin{aligned}
 f(x) &= \frac{1}{2 \cdot x - 5}, f(0) = -\frac{1}{5} \\
 f'(x) &= -\frac{2}{(2 \cdot x - 5)^2}, f'(0) = -\frac{2}{25} \\
 f''(x) &= \frac{8}{(2 \cdot x - 5)^3}, f''(0) = -\frac{8}{125} \\
 f'''(x) &= -\frac{48}{(2 \cdot x - 5)^4}, f'''(0) = -\frac{48}{625} \\
 f^{(4)}(x) &= \frac{384}{(2 \cdot x - 5)^5}, f^{(4)}(0) = -\frac{384}{3125} \\
 f^{(5)}(x) &= -\frac{3840}{(2 \cdot x - 5)^6}, f^{(5)}(0) = -\frac{768}{3125}
 \end{aligned}$$

$f(x) = \frac{1}{2x-5}$, а фиолетовым обозначим функция ряда:

Количество членов ряда	График
3	
4	
5	



Как мы можем заметить, при увеличении членов ряда Тейлора, график двух функций совпадает в большем количестве точек, чем при сумме меньшего количества членов. Тем самым можно сказать, что при увеличении кол-ва членов, функция суммы ряда приближается к значению функции, как я сказал ранее. Необходимо выяснить, при каком количестве членов ряда, их сумма достигнет значения $f(x)$ в некоторой точке $x \in [a, b]$, где $[a, b] = [0, 2]$ (дано в варианте)

Алгоритм

Вычисляем значение машинного эпсилона:

```
double FEps (void)
{
    double eps = 1.0;
    while (eps / 2.0 + 1.0 > 1.0) {
        eps /= 2.0;
    }
    return eps;
}
```

Машинный эпсилон — числовое значение, меньше которого невозможно задавать относительную точность для любого алгоритма, возвращающего вещественные числа. Абсолютное значение «машинного эпсилон» зависит от разрядности сетки применяемой ЭВМ, типа (разрядности) используемых при расчетах чисел, и от принятой в конкретном трансляторе структуры представления вещественных чисел (количества бит, отводимых на мантиссу и на порядок). Формально машинный эпсилон обычно определяют как минимальное из чисел ϵ , для которого $1+\epsilon > 1$ при машинных расчетах с числами данного типа. Альтернативное определение — максимальное ϵ , для которого справедливо равенство $1+\epsilon=1$.

Введем значение k и количество разбиений отрезка $[a, b]$ с клавиатуры.

Запустим цикл перебора аргумента функций от начало отрезка до его конца с шагом, равным длине отрезка $[a, b]$, деленным на значение разбиения отрезка.

Запустим цикл вычисления суммы ряда Тейлора, условием завершения которого является выполнение 100 операций либо если $|f(x) - ftaylor(x)| < \varepsilon * k$, то есть если разница между значениями меньше некоторого значения точности, умноженного на значение машинного эпсилона (x - аргумент функций).

Выводим в формате таблицы следующие значения:

| значение аргумента (x) | $ftaylor(x)$ | $f(x)$ | количество итераций |

Для удобства будем выводить значений аргумента функции с точностью до 2 знака после запятой, а значения функций с точностью до 10 знака.

После вывода каждой строки таблицы, обнуляем счетчик итераций и значение суммы ряда.

Список переменных

Название переменной	Описание
double a	Начало отрезка $[a, b]$
double b	Конец отрезка $[a, b]$
int k	Коэффициент точности
int n	Количество разбиений отрезка
int countIter	Счетчик количества итераций ряда Тейлора
double eps	Значение машинного эпсилона
double sum	Сумма n -ых членов ряда Тейлора
double Taylor	Формула вычисления члена ряда Тейлора

Список функций

Название функции	Описание
double FEps	Функция, вычисляющая значение машинного эпсилона
double Function	Функция, вычисляющая значение обычным методом от аргумента

Общие сведения о программе

Программное обеспечение	Xcode
-------------------------	-------

Аппаратное обеспечение	Macbook Pro 14` M1 Pro
Операционная система	macOS Montera
Язык программирования	C
Кол-во строк программы	67
Местонахождение файла	/Users/Ivan/Desktop/labs/kp3
Способ вызова	Компиляция в Xcode

Код программы

```

/*
    Ляпин Иван Алексеевич
    М80-104Б-22
    КП 3
    Вариант 8
*/

#include <math.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <string.h>
#include <stdbool.h>

double FEps (void) // функция для вычисления машинного эпсилона
{
    double eps = 1.0;
    while (eps / 2.0 + 1.0 > 1.0) {
        eps /= 2.0;
    }
    return eps;
}

double Function (double x) // функция, вычисляемая обычным методом
{
    return (1 / (2 * x - 5)); // выводим значение функции
}

int main (int argc, const char * argv[])
{
    double a = 0.0; //начало отрезка
    double b = 2.0; //конец отрезка
    int k, n; // k – коэффициент точности, n – количество раделенных отрезков
    double eps = FEps(); // значение машинного эпсилона
    int countIter = 0; // количество итераций
    double FTaylor; // функция Тейлора
    double sum = 0.0; // сумма ряда Тейлора
    printf("Введите коэффициент точности и количество разбиений [a, b]\n");
    scanf("%d %d", &k, &n); // считываем значение коэффициента точности и
количество разделений
    printf("Машинное Эпсилон = %0.16lf\n", eps); // выводим значение eps с
точностью до 16 знака после точки
    printf("|-----|\n"); // чертим
структуру таблицы
    printf("| x | Taylor | 1 / (2x - 5) | Step |\n");
    printf("|-----|-----|-----|-----|\n");

```

```

    for (double x = a; x <= b; x += (b - a) / n) { // запускаем цикл прохождения
по отрезку с шагом, равным длине разделенного отрезка
        while (countIter < 100 || (fabs(Function(x)) - fabs(sum) < eps * k))
        { // цикл до 100 итераций, либо пока разница между значениями функции и функции
тейлора не будет меньше eps * k
            countIter += 1; // увеличиваем счетчик итерацию на единицу
            FTaylor = ((pow(2, countIter - 1) * (pow(x, countIter - 1))) /
pow(5, countIter)); // вычисляем член ряда
            sum -= FTaylor; // считаем сумму ряда
            if (fabs(Function(x)) - fabs(sum) < eps * k) { // если
                break;
            }
        }
        printf("| %0.2lf | %0.10lf | %0.10lf | %3d | \n", x, sum,
Function(x), countIter);
        countIter = 0;
        sum = 0;
    }

    printf(" |-----| \n"); // закрываем
таблицу
    return 0;
}

```

Результаты работы программы

Введите коэффициент точности и количество разбиений [a, b]

1 10

Машинное Эпсилон = 0.0000000000000002

x	Taylor	1 / (2x - 5)	Step
0.00	-0.2000000000	-0.2000000000	1
0.20	-0.2173913043	-0.2173913043	14
0.40	-0.2380952381	-0.2380952381	19
0.60	-0.2631578947	-0.2631578947	25
0.80	-0.2941176471	-0.2941176471	32
1.00	-0.3333333333	-0.3333333333	38
1.20	-0.3846153846	-0.3846153846	48
1.40	-0.4545454545	-0.4545454545	62
1.60	-0.5555555556	-0.5555555556	81
1.80	-0.7142857143	-0.7142857143	100
2.00	-0.9999999998	-1.0000000000	100

Program ended with exit code: 0

Введите коэффициент точности и количество разбиений [a, b]

2 5

Машинное Эпсилон = 0.0000000000000002

x	Taylor	1 / (2x - 5)	Step
0.00	-0.2000000000	-0.2000000000	1
0.40	-0.2380952381	-0.2380952381	19
0.80	-0.2941176471	-0.2941176471	31
1.20	-0.3846153846	-0.3846153846	47
1.60	-0.5555555556	-0.5555555556	79
2.00	-0.9999999998	-1.0000000000	100

Program ended with exit code: 0

Введите коэффициент точности и количество разбиений [a, b]

1 15

Машинное Эпсилон = 0.000000000000002

x	Taylor	1 / (2x - 5)	Step
0.00	-0.2000000000	-0.2000000000	1
0.13	-0.2112676056	-0.2112676056	12
0.27	-0.2238805970	-0.2238805970	16
0.40	-0.2380952381	-0.2380952381	19
0.53	-0.2542372881	-0.2542372881	23
0.67	-0.2727272727	-0.2727272727	27
0.80	-0.2941176471	-0.2941176471	31
0.93	-0.3191489362	-0.3191489362	36
1.07	-0.3488372093	-0.3488372093	42
1.20	-0.3846153846	-0.3846153846	48
1.33	-0.4285714286	-0.4285714286	57
1.47	-0.4838709677	-0.4838709677	68
1.60	-0.5555555556	-0.5555555556	81
1.73	-0.6521739130	-0.6521739130	100
1.87	-0.7894736842	-0.7894736842	100
2.00	-0.9999999998	-1.0000000000	100

Program ended with exit code: 0

Вывод

Исходя из полученных результатов, можно заметить, что в некоторых случаях значения функций различаются уже на 10-ом знаке после запятой. Также можно сформулировать необходимость использования разложения функции в степенной ряд, точнее для чего это нужно и где это может быть использовано. Если взглянуть на таблицу, где графически представлены функции $f(x)$ и сумма нескольких членов ряда Тейлора, можно заметить, что с добавлением каждого следующего члена удается предугадать поведение графика, тем самым благодаря разложению функции в ряд, получается упростить исследование функции, сведя работу к изучению более простых элементов. Нельзя назвать данный метод эффективным, так как при работе гораздо проще воспользоваться уже встроенными функциями, так как это удобнее и затрачивает меньше времени. Но с точки зрения математики это весьма полезный способ исследования функций. Благодаря данному курсовому проекту, я закрепил свои знания в области математического анализа, а также научился применять их в программировании. Если бы я проводил расчеты вручную, то потратил куда больше времени чем при написании программы.

Дополнительное задание

Вычислим значения машинного эпсилона на языке Си для разных типов данных и сравним их со встроенными значениями:

Программа:

```

/*
    Ляпин Иван Алексеевич
    М80-1045-22
    КП 3
    Вариант 8
*/

#include <math.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <string.h>
#include <stdbool.h>
#include <float.h>

int main (int argc, const char * argv[])
{
    float eps_float = 1; //eps для типа float
    double eps_double = 1; //eps для типа double
    long double eps_ldouble = 1; //eps для типа long double
    while (eps_float + 1 > 1) { //вычисляем эпсилон
        eps_float /= 2;
    }
    while (eps_double + 1 > 1) {
        eps_double /= 2;
    }
    while (eps_ldouble + 1 > 1) {
        eps_ldouble /= 2;
    }

    printf("-----\n");
    printf("| Метод вычисления | float epsilon | double epsilon | long double epsilon |\n");
    printf("| Вычисленный | %.25f | %.25f | %.25Lf|\n",
    eps_float, eps_double, eps_ldouble);
    printf("| Константа ЯП | %.25f | %.25f | %.25Lf|\n",
    FLT_EPSILON, DBL_EPSILON, LDBL_EPSILON);
    return 0;
}

```

Вывод программы:

```

-----
| Метод вычисления | float epsilon | double epsilon | long double epsilon |
| Вычисленный | 0.0000000596046447753906250 | 0.000000000000000110223025 | 0.000000000000000110223025 |
| Константа ЯП | 0.0000001192092895507812500 | 0.0000000000000002220446049 | 0.0000000000000002220446049 |

```

Вычислим значение машинного эпсилон на языке Python:

```

import sys

eps = 1.0

while eps + 1.0 > 1.0:
    eps /= 2

print(' |    Вычисленный эpsilon    |    Встроенный эpsilon    | ')
print(' |', float(eps), ' |', sys.float_info.epsilon, '|')

```

Вывод программы:

	Вычисленный эpsilon		Встроенный эpsilon	
	1.1102230246251565e-16		2.220446049250313e-16	

Справочник

1 - https://ru.wikipedia.org/wiki/Ряд_Тейлора

2 - <https://www.mathnet.ru/links/0122300a1176aebfbbd62501f82520ae/nano712.pdf>