МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Курсовой проект
по курсу «Основы информатики»
I семестр

Задание 3
«Вещественный тип. Приближенные вычисления. Табулирование функции»

Группа: М80 – 107Б-18

Студент: Цапков Александр Максимович

Преподаватель: Ридли Александра Николаевна

Оценка: ____________________

Дата:_____________________

Москва, 2018

Содержание

Введение

Третье задание курсового проекта состоит в том, чтобы составить программу на си которая печатает таблицу значений элементарной функции вычисленной двумя способами: по формуле Тейлора и с помощью встроенных функция языка программирования.

Основная концепция вычисления

Функция, которую я буду считать: $\frac{3x-5}{x^2-4x+3}$

Это достаточно простая функция и чтобы ее посчитать даже не нужно подключать библиотеку math.h. Формула для расчета значения функции в точка просто будет выглядеть так:

F = (3 * x - 5) / (x * x - 4 * x + 3)

Теперь нам осталось выяснить как считать значение функции по формуле Тейлора. Разложение этой функции уже дано в самом задании лабораторной, что значительно упрощает задачу. Мы просто должны посчитать несколько первых членов разложения. Но до какого члена считать? Здесь нам пригодится машинное эпсилон.

Машинное эпсилон — числовое значение, меньше которого невозможно задавать относительную точность для любого алгоритма, возвращающего вещественные числа. Для его расчета воспользуемся рекурсивной функцией, в основе которой лежит данной сравнение:

```
1.0 + e / 2.0 > 1.0
```

Если e/2 становится настолько мало, что его добавление к единице ничего не меняет, значит сам e — это и есть машинное эпсилон.

Где же его используем мы? Как только разница между двумя соседними членами становится меньше чем епсилон, можно прекратить вычисление следующих членов, так как это уже не будет иметь смысла.

Подбор дельта разниц членов ряда Тейлора

Приступим к расчету членов ряда Тейлора. Но как именно их считать? Просто считать по формуле абсолютно неэффективно, так как снизу под знаком деления находится факториал числа равного номеру члена. Уже на 20-м шаге компьютер не сможет корректно посчитать значение члена последовательности. Нужно применить другой метод.

И такой метод есть. Каждый следующий член формулы Тейлора отличается от предыдущего на какой-то коэффициент. Если найти этот коэффициент, то можно каждый шаг просто домножать предыдущий член на этот коэффициент. Экспериментально я вывел такой коэффициент:

```
currentStepValue = lastSyepValue * x / 3.0 + (2.0 / 3.0) * power(x, step);
```

Но в этом случае возникает небольшая проблема. Возведение в степень является достаточно неточной, и в идеале хорошо от этого избавится. Моя идея заключается в том, чтобы разбить один член ряда Тейлора на как бы 2, и каждый из них домножать на свой коэффициент, тогда можно избавится от степени в формуле.

```
currentStepValue_11 = lastSyepValue_11 * x;
currentStepValue_12 = lastSyepValue_12 * x / 3.0;
```

Вывод результатов

Теперь осталось при расчете каждого нового члена просто добавлять его к сумме этих членов, что и будет являться значением функции. В проверке разниц между членами можно вставить переменную k, которая будет задавать точность наших вычислений.

```
absR(currentStepValue - lastSyepValue) > e * k
```

Таким образом чем больше число k, тем меньше точность вычислений, и при к разным 1 она максимальна.

Для сравнения результатов я решил вывести значение посчитанной с помощью первого способа функции Тейлора и с помощью второго (более эффективного) для проверки влияния функции степени на точность результата. Также я выведу количество итераций, после которых дельта членов стала меньше эпсилон для каждого из способов. И вот она получившаяся таблица.

```
Alexs-MacBook-Air:3 alex$ gcc -std=c99 -Wall -pedantic -g  main.c
Alexs-MacBook-Air:3 alex$ ./a.out
11 1
e = 0.0000000000000002220446049
n = 11
              x                                f(x)
0.0000000000000000000000000 -1.6666666666666667406815350
0.0500000000000000027755576 -1.7305976806422833647047810
0.1000000000000000055511151 -1.8007662835249043098428956
0.1499999999999999944488849 -1.8782249742002063985779614
0.2000000000000000111022302 -1.9642857142857141905523122
0.2500000000000000000000000 -2.0606060606060605522316109
0.2999999999999999888977698 -2.1693121693121688586813889
0.3499999999999999777955395 -2.2931785195936136290129070
0.4000000000000000222044605 -2.4358974358974356810847439
0.4500000000000000111022302 -2.6024955436720142998296978
0.5000000000000000000000000 -2.7999999999999998223643161

          taylor f(x)                    taylor f(x) v.2           iter
-1.6666666666666665186369300 -1.6666666666666665186369300   3    3
-1.7305976806422835867493859 -1.7305976806422835867493859   15   15
-1.8007662835249040877982907 -1.8007662835249040877982907   18   18
-1.8782249742002063985779614 -1.8782249742002063985779614   21   21
-1.9642857142857137464631023 -1.9642857142857137464631023   25   25
-2.0606060606060596640531912 -2.0606060606060596640531912   28   28
-2.1693121693121697468598086 -2.1693121693121697468598086   32   32
-2.2931785195936131849236972 -2.2931785195936131849236972   36   36
-2.4358974358974347929063242 -2.4358974358974347929063242   41   41
-2.6024955436720134116512781 -2.6024955436720134116512781   47   47
-2.7999999999999989341858964 -2.7999999999999998223643161   53   53
```

Заключение (анализ результатов)

Как мы можем видеть разница между высчитанными нами значениями по формуле Тейлора и с помощью средств языка Си различаются не больше чем на эпсилон. Также разница между значениями 2-х способов нахождения значения по Тейлору небольшая и количество итераций идентично, но все же можно заметить, что результат второго, более эффективного способа чуть ближе к результату обычного вычисления функции, что говорит о том, что предположение о неэффективности функции степени было верным.

Еще одно интересное наблюдение связано с количеством итераций расчета по формуле Тейлора. С увеличением аргумента росло и количество итераций. Это происходит из-за так называемого радиуса схождения. Формула Тейлора приблизительно равна самой функции в неком радиусе вокруг точки по которой мы разложили эту функцию. В этой точке значение наиболее сходится с самой функцией и не требуется большого количества итераций. Но по мере приближения к краю радиуса схождения требуется все больше итераций формулы Тейлора, чтобы ее значения совпадали со значением самой функции.

Источники

1. https://toster.ru/q/160995 - Машинный эпсилон
2. https://youtu.be/3d6DsjIBzJ4 - Ряд Тейлора | Сущность матанализа, глава 11