Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет «Прикладная математика и физика»

Кафедра «Вычислительная математика и информатика»

Курсовой проект по

Информатике и вычислительной технике по теме: «Вещественный тип. Приближенные вычисления. Табулирование функций»

Выполнил: Щербаков А.А.

Студент группы М8О-106Б

Преподаватель: Дубинин А.В.

Оценка:

Дата:

Москва 2016

Введение

В реальном мире нам приходится сталкиваться с множеством весьма прагматичных задач. Например, нам нужно посчитать ту же экспоненту или синус для различных значений аргумента x. Для чего? Экспонента очень хорошо описывает рост популяции микроорганизмов в насыщенной кормом среде, синус и косинус дают прекрасное описание волновых процессов... А все это очень пригодится, когда мы будем строить космические корабли, изобретать вакцины... Конечно, сейчас посчитать различные трансцендентные функции могут практически любые инженерные калькуляторы, но не всегда функция может быть напрямую обработана вычислительной машиной (да и на данный момент нам нужно просто понять саму идею). Тогда ставится задача написания алгоритма, который будет состоять из более простых операций, и который все-таки позволит нам выполнить желаемые действия с функцией. Одним из таких алгоритмов является использование формулы Тейлора.

Формула Тейлора

Рядом Тейлора функции f(x), дифференцируемой в точке a n+1 раз называется функциональный формальный ряд:

$\sum_{i=0}^{k}\left(\frac{f^{(k)}(a)}{k!}\right)(x-a)^{k}$, где a – точка, в окрестности которой ведется разложение, R_{n+1} – остаточный член формулы Тейлора.

Данная формула позволяет привести функцию, дифференцируемую достаточное количество раз, к алгебраическому многочлену. Таким образом, компьютер, который может складывать и умножать числа, способен вычислить значение трансцендентных, дробно-рациональных, и, вообще говоря, других дифференцируемых функций. Этот алгоритм не является наиболее точным, быстрым и эффективным, но служит хорошим примером использования простых операций вместо сложных.

Вычисление значения функции $y = 3^x$ с помощью формулы Тейлора

Разложение по формуле Тейлора для $\mathbf{3^x}$:

$$3^x = 1 + \frac{\ln(3)}{1!}x \quad + \frac{ln^2(3)}{2!}x^2 + \cdots + \frac{ln^n(3)}{n!}x^n$$

Данная формула предполагает вычисление значений в окрестности нуля, поэтому при увеличении расстояния от нуля до точки, в которой мы ищем значение, погрешность вычислений будет расти.

Определение основных положений:

- Для вычисления максимально точного значения функции 3^x будем использовать библиотеку math.h: $\mathbf{3^x}$ **:= pow(3, x)** , (orig – значение функции в точке x);
- Значение, полученное по разложению: **taylor** ;
- Погрешность: **delta** = **orig – taylor**;
- Тип данных, с которым мы работаем, ограничивает точность вычисляемого значения функции, поэтому имеет смысл ограничить число вычисляемых одночленов из формулы Тейлора. Таким образом, программа прекратит вычисления как только значение функции в точке перестанет меняться. Величина **iters** будет показывать, после какого значения **n** прграмма прекратит расчеты.
- Машинное эпсилон (**eps**) — минимальное число, отличное от нуля, при прибавлении которого к единице получается число, отличное от единицы. Зависит от используемого типа данных и разрядности машины. Именно благодаря этой величине мы поймем, в какой момент проводить дальнейшие расчеты бессмысленно.

Описание работы программы

1. Вычисление машинного эпсилон;
2. Считывание со стандартного ввода числа, указывающего количество разбиений на отрезке (**cnt**);
3. Вычисление расстояния между соседними точками после разбиения – шага.
4. Вычисление значения функции в текущей точке по формуле Тейлора: пока значение одночлена (полученного при определенном n) больше эпсилон, суммировать данные значения;
5. Вычисление значения функции в текущей точке при помощи стандартных функций из библиотеки math.h;
6. Вычисление погрешности;
7. Переход в следующую точку
8. Выполнить пункты 2-7 **cnt** раз.

Заключение

Таким образом, сведение трансцендентной функции к алгебраическому многочлену позволяет вычислить относительно точное значение функции в точке. Очевидно, что абсолютной точности для всех расчётов мы не добьёмся никогда, но чаще всего для практических целей нам такое излишество и ни к чему. Пять, шесть знаков после запятой - этого бывает более, чем достаточно, когда речь идёт о проектировке сложного технического устройства (а обычно и того меньше). Так что данный метод спокойно может быть применен в различных областях науки.