Московский Авиационный Институт

(национальный исследовательский университет)

Факультет «Прикладной математики и информатики»

Кафедра 806

Курсовой проект по теме: «Вещественный тип. Приближенные вычисления. Табулирование функции»

Выполнил: студент 1 курса

Каширин К.Д.

Преподаватель: Трубченко Н.М.

Москва 2020

Содержание

Задача

Составить программу на Си, которая печатает таблицу значений элементарной функции, вычисленной двумя способами: по формуле Тейлора и с помощью встроенных функций языка программирования. В качестве аргументов таблицы взять точки разбиения отрезка [a, b] на n равных частей (n+1 точка, включая концы отрезка), находящихся в рекомендованной области хорошей точности формулы Тейлора. Вычисления по формуле Тейлора проводить по экономной в сложностном смысле схеме с точностью e*k, где e — машинное эпсилон аппаратно реализованного вещественного 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 e и обеспечивать корректные размеры генерируемой таблицы.

Определение машинного эпсилон

Машинный эпсилон - это минимальная разница между числами, которую компьютер в состоянии различить. Если два вещественных числа A и B отличаются меньше чем на машинный эпсилон, компьютер их не различит, и будет в своей памяти представлять каким-то третьим числом C, таким что

$$\begin{cases} |A-B| < \epsilon \\ |A-C| < \epsilon \\ |B-C| < \epsilon \end{cases}$$

Чтобы рассчитать значение ε можно использовать следующую функцию:

```
double macheps()
{
        double e = 1.0;
        while (1.0 + e / 2.0 > 1.0)
                e /= 2.0;
        return e;
}
```

Э

Идея

С помощью функции macheps вычислить машинный эпсилон и использовать его для сравнения значений, вычисленных по формуле Тейлора и функцией, реализованной самостоятельно. Для вычисленных значений можно использовать массив структур Point с двумя полями — x и y. Память на массив выделяется динамически.

Функции taylor_function на вход подается значение n (порядок до которого следует высчитывает значение последовательности, значение x).

Функции function на вход подается значение x, возвращается значение функции $\mathbf{2} * (\cos \boldsymbol{x})^{\mathbf{2}} - \mathbf{1}$

Метод calculation принимает количество итераций, левую и правую границы отрезка, по которому нужно итерироваться, и два указателя: на функции taylor_function и function. Пока разность между посчитанными значениями не станет меньше $\varepsilon * k$(в нашем случае $k = 100$) , нужно продолжать вычисления. Максимальное число повторений вычислений — 100.

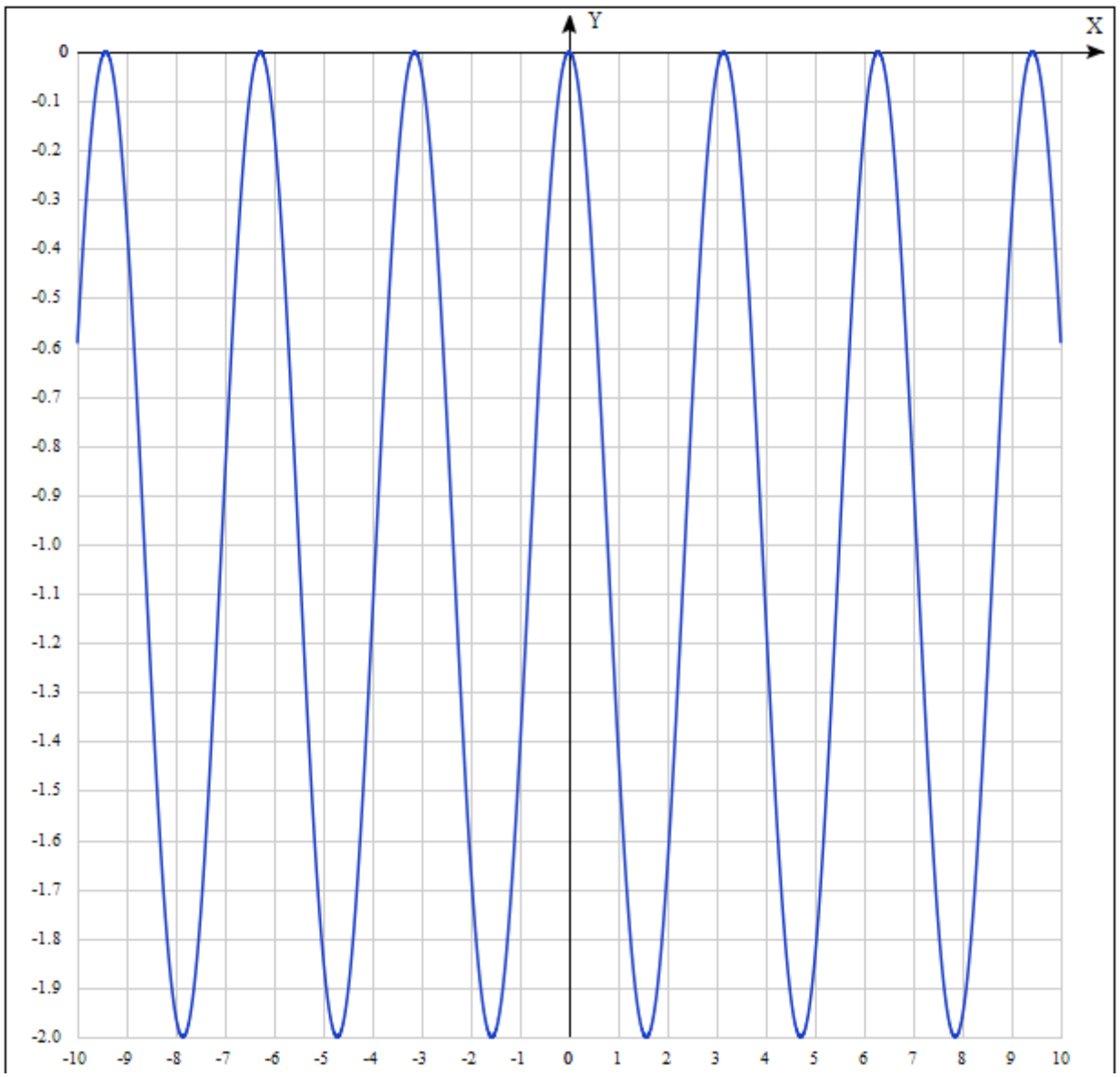
Вариант 5

5	$-\frac{4x^2}{2}+\frac{16x^4}{24}+\ldots+(-1)^n\frac{(2x)^{2n}}{(2n)!}$	0.0	0.5	$2(\cos^2 x-1)$

Протокол программы:

```
kirill@LAPTOP-F153AKTP:~/k$ gcc kurs_var5.c -lm
kirill@LAPTOP-F153AKTP:~/k$ ./a.out
10
0| 0.000000 0.000000 0.000000
1| 0.050000 -0.004996 -0.004996
2| 0.100000 -0.019933 -0.019933
3| 0.150000 -0.044664 -0.044664
4| 0.200000 -0.078939 -0.078939
5| 0.250000 -0.122417 -0.122417
6| 0.300000 -0.174664 -0.174664
7| 0.350000 -0.235158 -0.235158
8| 0.400000 -0.303293 -0.303293
9| 0.450000 -0.378390 -0.378390
10| 0.500000 -0.459698 -0.459698
```

График функции:



Полный текст программы:

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

double macheps()
{
    double e = 1.0;
    while (1.0 + e / 2.0 > 1.0)
        e /= 2.0;
    return e;
}


struct Point {
    double x;
    double y;
};
long long factorial(int n){
    long long result=1;
    for (int i=1;i<=n;i++){
        result = result*i;
    }
    return result;
}
double taylor_function(unsigned n, double x)
{
    double y = 0.;
    for(unsigned i = 1; i <= n; ++i) {
        if (i % 2 == 0){
            y = y + pow(2*x,2*i)/(factorial(2*i));
        }
        else{
            y = y + (-pow(2*x,2*i))/(factorial(2*i));
        }
    }
    return y;
}
double function(double x)
{
    return 2*(cos(x)*cos(x)-1);
}


void calculate(unsigned iterationCount, double a, double b, double (*taylor_f)(unsigned, double), double (*real_f)(double))
{
    double step = ( b - a ) / iterationCount;
    struct Point* points = (struct Point*)malloc(sizeof(struct Point) * (iterationCount+1));
    double eps = macheps();
    double x = a;
    points[0].x = x;
    points[0].y = 2*(cos(x)*cos(x)-1);
    printf("%d| %lf %lf %lf\n", 0, x, real_f(x), points[0].y);
    x+=step;
    for(unsigned i = 1; i <= iterationCount; ++i, x+=step) {
        for (unsigned p = 1;(p<100) && (fabs(real_f(x) - taylor_f(p, x)) > eps * 100);p++){
            points[i].x = x;
            points[i].y = taylor_f(p, x);
        }
        printf("%d| %lf %lf %lf\n", i, x, real_f(x), points[i].y);
    }
}

int main()
{
    unsigned n;
    double a = 0.0, b = 0.5;
    scanf("%u", &n);
    calculate(n, a, b, taylor_function, function);
}
```

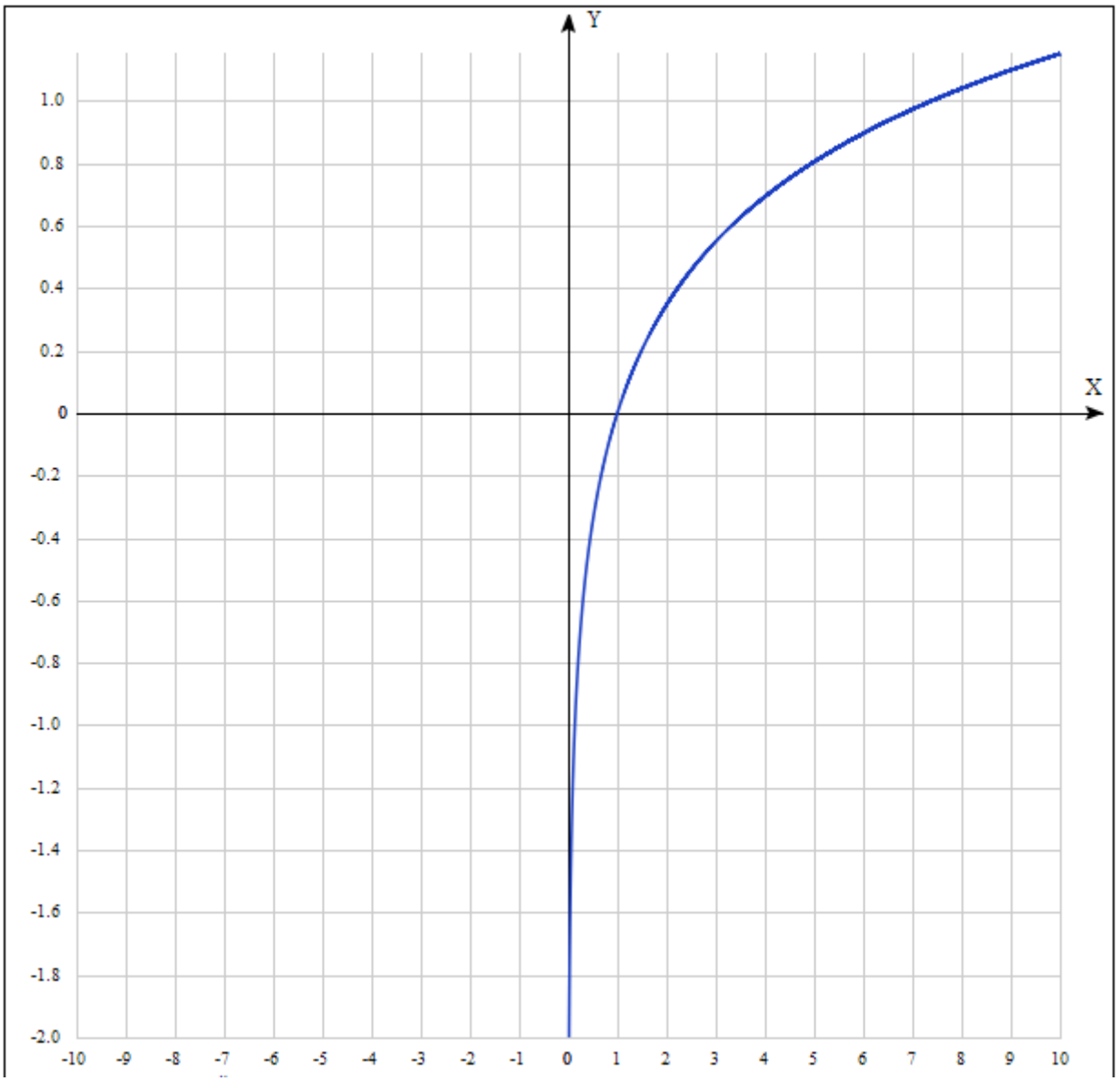
Вариант 17

17	$\frac{x-1}{x+1}+\frac{1}{3}(\frac{x-1}{x+1})^3+\ldots+\frac{1}{2n+1}(\frac{x-1}{x+1})^{2n+1}$	0.2	0.7	$\frac{1}{2}\ln x$

Протокол программы:

```
kirill@LAPTOP-F153AKTP:~/k$ gcc kurs_var17.c -lm
kirill@LAPTOP-F153AKTP:~/k$ ./a.out
10
0| 0.200000 -0.804719 -0.804719
1| 0.250000 -0.693147 -0.693147
2| 0.300000 -0.601986 -0.601986
3| 0.350000 -0.524911 -0.524911
4| 0.400000 -0.458145 -0.458145
5| 0.450000 -0.399254 -0.399254
6| 0.500000 -0.346574 -0.346574
7| 0.550000 -0.298919 -0.298919
8| 0.600000 -0.255413 -0.255413
9| 0.650000 -0.215391 -0.215391
10| 0.700000 -0.178337 -0.178337
```

График функции:



Полный текст программы:

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

double macheps()
{
    double e = 1.0;
    while (1.0 + e / 2.0 > 1.0)
        e /= 2.0;
    return e;
}


struct Point {
    double x;
    double y;
};

double taylor_function(unsigned n, double x)
{
    double y = 0.;
    for(unsigned i = 0; i <= n; ++i) {
        y=y+pow((x-1)/(x+1),2*i+1)/(2*i+1);
    }
return y;
}
double function(double x)
{
    return 0.5*log(x);
}


void calculate(unsigned iterationCount, double a, double b, double (*taylor_f)(unsigned, double), double (*real_f)(double))
{
    double step = ( b - a ) / iterationCount;
    struct Point* points = (struct Point*)malloc(sizeof(struct Point) * (iterationCount+1));
    double eps = macheps();
    double x = a;
    points[0].x = x;
    points[0].y = 0.5*log(x);
    printf("%d| %lf %lf %lf\n", 0, x, real_f(x), points[0].y);
    x+=step;
    for(unsigned i = 1; i <= iterationCount; ++i, x+=step) {
        for (unsigned p = 1;(p<100) && (fabs(real_f(x) - taylor_f(p, x)) > eps * 100);p++){
            points[i].x = x;
            points[i].y = taylor_f(p, x);
        }
        printf("%d| %lf %lf %lf\n", i, x, real_f(x), points[i].y);
    }
}

int main()
{
    unsigned n;
    double a = 0.2, b = 0.7;
    scanf("%u", &n);
    calculate(n, a, b, taylor_function, function);
}
```

Вывод

Я составил программу на Си, получил точные значения вычислений данных функций.