计算方法第二次上机作业程序文档

5.6.2021 yawning-lion

一、任务介绍

给定一个椭圆的半长轴和半短轴，计算近似椭圆周长，给定误差限，要求使用Romberg求积法，计算结果误差在误差限内。

二、公式说明

1、被积函数

给定椭圆

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$$

求椭圆周长的问题，实际上转化为一个积分

$$l=2\int_{-a}^{a}\sqrt{1+(\frac{y'}{x'})^2}dx$$

为便于计算，使用三角换元

$$\begin{cases} x=acos\theta \\ y=bsin\theta \end{cases}$$

积分化为

$$l=4\int_{0}^{\frac{\pi}{2}}\sqrt{b^2+(a^2-b^2)sin^2\theta}d\theta$$

故实际上考虑被积函数

$$f(x)=4\sqrt{b^2+(a^2-b^2)sin^2x},[0,\frac{\pi}{2}]$$

的积分。

2、复化梯形积分

将区间2等分时，复化梯形公式为

$$I=\frac{h}{2}(f(a)+f(b))$$

当把区间加细为n份时，由复化梯形求积公式

$$I=\frac{h}{2}(f(a)+f(b)+2\sum_{i=1}^{n-1}f(a+ih))$$

其中$a=0,b=\frac{\pi}{2},h=\frac{b-a}{n}$。

3、Romberg外推

依次地，给定k时，对于$m=1,2...k$,利用公式

$$T_m^{(k-m)}=\frac{4^mT_{m-1}^{(k-m+1)}-T_{m-1}^{(k-m)}}{4^m-1}$$

不断延展外推序列。

三、程序说明

1、运行环境

程序编译环境为mingw-w64-v8.0.0,g++,IDE为vscode，程序文档利用markdown写作。

2、使用说明

- **输入规范**

  本程序需要用户输入的数据有三个。

  - 在程序打印`please enter the value of the majorAxis:`后，输入目标椭圆的**半长轴**大小，数据类型限制为双精度数。例如2。

- 在程序打印`please enter the value of the minorAxis:`后，输入目标椭圆的**半短轴**大小，数据类型限制为双精度数。例如1。
- 在程序打印`please enter the value of the allowable error:`后，输入数值积分结果的**容许误差**大小，数据类型限制为双精度数。例如，要求积分结果具有5位有效数字，则容许误差为0.00005.

- **输出格式**

  完成计算后，将结果输入进与代码同文件夹的output.txt中，分别输出外推次数为m时的积分值和最终满足精度要求的积分值。

3、程序结构

本程序包含三个头文件

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

开始前，定义常量

```
#define MAXNUM 64
const double PIE=3.1415926535897932384626433832‬8;
const double A=0;
const double B=PIE/2;
```

`MAXMUM`为之后存储外推积分序列的二维数组的最大长、宽值；`A` `B`则是积分区间的左右边界。

随后定义存储椭圆半长短轴的结构体和相应指针

```
typedef struct Oval
{
    double majorAxis;
    double minorAxis;
}COval;
typedef COval* pCOval;
```

随后定义一众函数。

主程序中，先完成所有输入，再利用输入，调用函数`IniOva(majorAxis,minorAxis)`对目标椭圆`oval`进行初始化。

随后定义存储所有Romberg积分过程中所需积分值的数组`integralT[MAXNUM][MAXNUM]`，并计算将区间分为2份时的复化梯形积分值，即`integralT[0][0]`.

下面进入由变量`k`控制的`do-while`循环。

每次循环开始时都计算当区间分为2^{k+1}时的复化梯形积分值，即`integralT[0][k]`的值，下面调用函数`Lengthen_Tm(k,integralT,EPSILON)`，利用新计算出的复化梯形积分值进行新的外推，根据函数返回值`state`来判断：

- 若`state`为负数，则代表这一轮外推中没有积分值误差小于等于容许误差，需要进入下一次循环再次外推。
- 若`state`不为负数，则代表这一轮外推中出现了在容许误差范围内的积分值，函数返回值`state`为这个积分值。

循环在`state`不为负数时退出，进行输出，程序结束运行。

4、函数说明

`double OvalFx(double Theta,pCOval oval)`

事实上给出了自变量为θ时的被积函数的计算式，**输入**`Theta`为自变量值，`oval`为指向储存目标椭圆所有信息结构体的指针，**返回**被积函数在这个自变量下的取值。

`pCOval IniOva(double majorAxis,double minorAxis)`

输入椭圆的半长轴`majorAxis`半短轴`minorAxis`.

返回储存这些信息的结构体的指针。

`int Lengthen_T0(int k,double h,int n,double integralT[][MAXNUM],pCOval oval)`

输入需要计算的复化梯形积分在序列中的位置`k`，积分步长`h`，区间份数`n`，存储所有积分序列的二维矩阵指针`integralT[][MAXNUM]`,目标椭圆`oval`。

计算$T_0^{(k)}$并将`integralT[0][k]`更新为相应值。完成复化梯形积分序列的伸长。

返回值无意义。

`double CalErr(double a,double b)`

输入`a` `b`，计算差的绝对值并**返回**。

`double Lengthen_Tm(int k,double integralT[][MAXNUM],double EPSILON)`

输入`k`，实际上利用`k`得到各积分序列中需要更新的位置；**输入**存储所有积分序列的二维矩阵指针`integralT[][MAXNUM]`,容许误差值`EPSILON`。

对于m依次取$1, 2...k$，利用$T_{m-1}^{(k-m)}$和$T_{m-1}^{(k-m+1)}$计算$T_m^{(k-m)}$并更新`integralT[m][k-m]`为相应值。不断延展外推序列。过程中，计算$T_m^{(0)}$和$T_{m-1}^{(0)}$的差值作为此时数值积分的误差，当误差不大于容许误差值时直接**返回**$T_m^{(0)}$即`integralT[m][k-m]`作为积分结果。否则**返回**`-1`作为这一轮外推仍未得到满足精度要求的积分值，需要进入下一轮外推。

三、算例展示

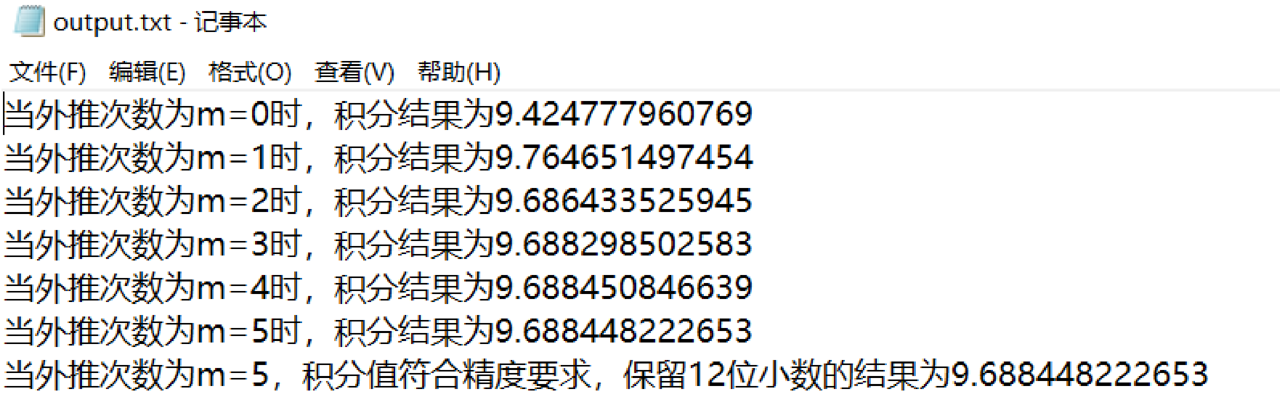
输入半长轴半短轴分别为2，1，即计算椭圆

$$\frac{x^2}{4} + y^2 = 1$$

的周长。

输入容许误差0.00005，（容易估计积分值小于10）即要求积分结果具有五位有效数字。

output.txt中输出结果为


output.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
当外推次数为m=0时，积分结果为9.424777960769
当外推次数为m=1时，积分结果为9.764651497454
当外推次数为m=2时，积分结果为9.686433525945
当外推次数为m=3时，积分结果为9.688298502583
当外推次数为m=4时，积分结果为9.688450846639
当外推次数为m=5时，积分结果为9.688448222653
当外推次数为m=5，积分值符合精度要求，保留12位小数的结果为9.688448222653

程序运行结果为9.688448222653,根据要求取为9.6884。

四、结果分析

应用Romberg积分法可以在不长的时间开销内将积分值精度提升到较高的值，但本程序的缺点是空间利用率低，64×64的二维数组利用率最高只有一半，实际上定义为10×10的数组或许更为合理，因为一般外推次数为6时就可以达到可观的精度，再继续外推对精度的提升也不大。