

天津理工大学 实验报告

学院（系）名称：计算机科学与工程学院

姓名	王帆	学号	20152180	专业	计算机科学与技术
班级	15 计算机 1 班	实验项目	实验三 数值积分		
课程名称		数值计算方法		课程代码	0665026
实验时间		2017 年 5 月 22 日 第 7 - 8 节		实验地点	7-220、7-219
批改意见				成绩	
教师签字：					

一、 实验目的

掌握复化求积法和龙贝格算法求积分近似值的方法，根据实验要求，使用这两种方法解决具体问题，设计相应的算法框图并编程实现，上机调试得到正确的运行结果。

二、 实验环境

- 硬件环境：IBM—PC 或兼容机
- 软件环境：Windows 操作系统，VC6.0
- 编程语言：C 或 C++

三、 实验内容

1. 用复化梯形公式计算下列积分值

$$\int_1^2 \frac{1}{x} dx$$

要求：

- (1) 区间的等分次数 n 由键盘终端输入；
 - (2) 绘制复化梯形公式求解积分的算法框图；
 - (3) 打印输出运行结果。
2. 用龙贝格算法计算下列积分值，使精确度达到 10^{-6} 。

$$\int_1^2 x^{\frac{3}{2}} dx$$

要求：

- (1) 绘制龙贝格算法的实现框图；

(2) 按照龙贝格算法的计算顺序按列依次打印输出梯形序列、辛普森序列、柯特斯序列和龙贝格序列的计算结果，如下所示；

k	区间等分数 $n=2^k$	梯形序列 T_2^k	辛普森序列 S_2^{k-1}	柯特斯序列 C_2^{k-2}	龙贝格序列 R_2^{k-3}
0	1	T_1			
1	2	T_2	S_1		
2	4	T_4	S_2	C_1	
3	8	T_8	S_4	C_2	R_1
4	16	T_{16}	S_8	C_4	R_2
5	32	T_{32}	S_{16}	C_8	R_4

(3) 输出最终满足精度要求的积分近似值。

四、 实验要求

1. 每一个实验内容要求自己独立完成，不允许抄袭别人，否则按不及格处理；
2. 按照实验要求，根据自己的程序编写情况绘制相应的算法框图或描述算法步骤；
3. 按照实验内容和相应的要求书写实验报告；
4. 在实验过程部分，要求根据实验内容和要求书写每一个实验相应的算法步骤或框图、运行过程和运行结果的截图、运行结果分析、以及程序源代码。每一个实验要求书写下述内容：

- (1) 算法步骤描述或算法框图
- (2) 程序源代码
- (3) 运行结果（要求截图）
- (4) 运行结果分析
5. 在规定的时间内上交实验报告。

五、 实验过程

1. 用复化梯形公式计算下列积分值

算法描述：

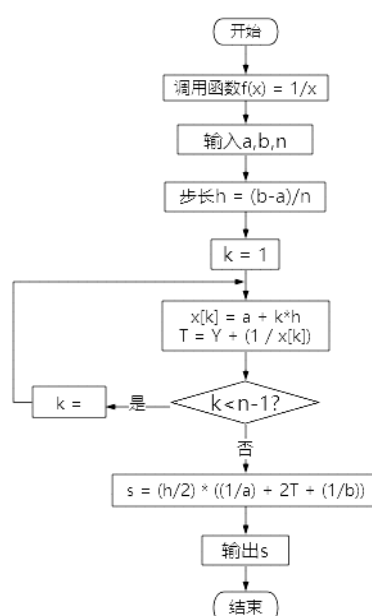
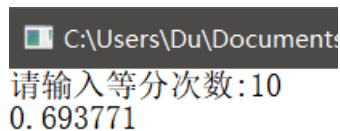


图 1-1 复化梯形法求解数值积分流程图

代码实现:

```
//复化梯形公式求解数值积分
#include<iostream>
#include<cmath>
using namespace std;
//全局变量
int n;
double a=1.0,b=2.0;
double h;
double sum=0.0,x=0.0;
//原型声明
void init();
void trapezium();
//实现
int main(){
    init();
    trapezium();
    return 0;
}
void init(){
    cout<<"请输入等分次数:";
    cin>>n;
}
void trapezium(){
    h=(b-a)/n;
    for(int i=1;i<=n-1;i++){
        x=a+i*h;
        sum+=1.0/x;
    }
    cout<<h/2.0*(1.0/a+2*sum+1.0/b);
}
```

运行结果:



```
C:\Users\Du\Documents
请输入等分次数:10
0.693771
```

图 1-2 复化梯形法求解数值积分

运行分析:

利用复化梯形公式求解数值积分,其精确度会随着等分次数的增加而增加。当等分次数趋近于无穷大时,即为精确解。

2. 用龙贝格算法计算下列积分值,使精确度达到 10^{-6} 。

算法描述:

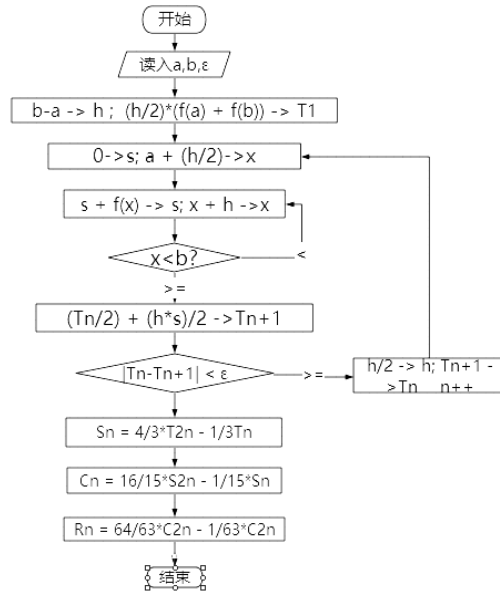


图 1-2 龙贝格算法求解数值积分流程图

代码实现:

//复化梯形公式求解数值积分

```
#include <iostream>
```

```
#include <cmath>
```

```
#include <iomanip>
```

```
using namespace std;
```

//全局变量

```
long double Tn[100],Sn[100],Cn[100],Rn[100];
```

```
double a=1.0,b=2.0;
```

```
double h;
```

```
double sum=0.0,xk=0.0;
```

```
int k;
```

```
bool temp=false;
```

//原型声明

```
void init();
```

```
void show(int i);
```

```
double trapezium(double T,int n);
```

```
double Simpson(double T1, double T2);
```

```
double Cotes(double S1, double S2);
```

```
double Romberg(double C1, double C2);
```

//实现

```
int main(){
```

```
    init();
```

```
    show(1);
```

```
    show(2);
```

```
    return 0;
```

```
}
```

```
void init(){
```

```
    Tn[1]=1.0/2*(pow(1.0,1.5)+pow(2.0,1.5));
```

```

    for(k=1;k<10;k++){
        Tn[k+1]=trapezium(Tn[k],pow(2,k-1));
    }
    for(k=1;k<10;k++){
        Sn[k]=Simpson(Tn[k+1],Tn[k]);
    }
    for(k=1;k<10;k++){
        Cn[k]=Cotes(Sn[k+1],Sn[k]);
    }
    for(k=1;k<10;k++){
        Rn[k]=Romberg(Cn[k+1],Cn[k]);
    }
}
void show(int i){
    switch(i){
        case 1:{
            cout<<"k"<<"    "<<"等分次数"<<
            "\t"<<"梯形序列"<<"\t"<<
            "辛普森序列"<<"\t"<<"柯特斯序列"<<
            "\t"<<"龙贝格序列"<<endl;
            break;
        }
        case 2:{
            for(k=0;k<10;k++){
                cout<<k<<"\t"<<pow(2,k)<<"\t"<<setprecision(8)<<Tn[k+1]<<"\t";
                if(k>=1) cout<<setprecision(8)<<Sn[k]<<"\t";
                else if(k>=2) cout<<setprecision(8)<<Cn[k-1]<<"\t";
                else if(k>=3){
                    cout<<setprecision(8)<<Rn[k-2];
                    if(Rn[k-2]-Rn[k-1]<0.000001&&temp){
                        break;
                    }
                    temp=true;
                }
                cout<<endl;
            }
            cout<<endl;
            cout<<"近似值为:"<<setprecision(7)<<Rn[k];
            break;
        }
    }
}
double trapezium(double T,int n){
    h=(b-a)/n;
    for(int i=0;i<=n-1;i++){

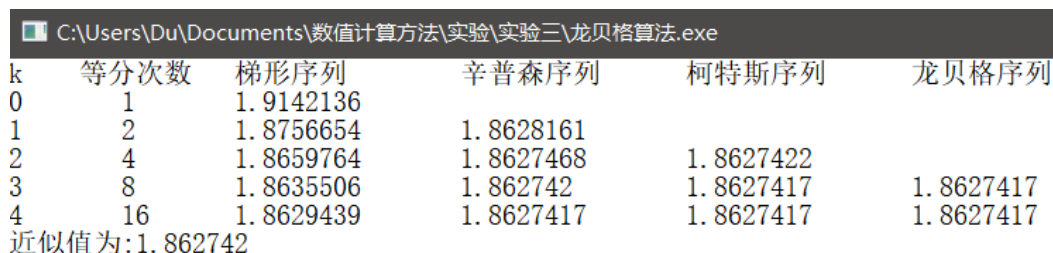
```

```

        xk=a+(i+1.0/2)*h;
        sum+=pow(xk,1.5);
    }
    return (h/2.0*sum+1.0/2*T);
}
double Simpson(double T1, double T2){
    return 4.0/3*T1-1.0/3*T2;
}
double Cotes(double S1, double S2){
    return 16.0/15*S1-1.0/15*S2;
}
double Romberg(double C1, double C2){
    return 64.0/63*C1-1.0/63*C2;
}

```

运行结果:



k	等分次数	梯形序列	辛普森序列	柯特斯序列	龙贝格序列
0	1	1.9142136			
1	2	1.8756654	1.8628161		
2	4	1.8659764	1.8627468	1.8627422	
3	8	1.8635506	1.862742	1.8627417	1.8627417
4	16	1.8629439	1.8627417	1.8627417	1.8627417

近似值为:1.862742

图 2-2 龙贝格算法运行结果

六、 实验总结及心得体会

通过本次实验，我加深了对数值积分的理解。除了最基本的梯形法，本实验还包括辛普森算法、柯特斯算法以及龙贝格算法。微积分学是高等数学最为关键的内容之一，通过学习数值微积分方法，我提升了对于利用计算机求解微积分问题近似值的能力，并有助于将它应用于未来的学习与科研中。