# 实习四实习报告

# 实习要求

**用数值积分方法近似计算**

****

**及圆周率**

****

**(a). 用复合Simpson求积公式计算，要求绝对误差限小于，试根据积分余项估计步长ℎ的取值范围.按要求选择一个步长进行计算，观察数值结果与误差要求是否相符.**

**(b). 用Romberg外推方法求积分近似值(误差要求与(a)同).**

**(c). 用下面的复合Gauss 公式计算近似积分：**

**其中ℎ=(𝑏−𝑎)/𝑛,𝑥𝑖+12=𝑥𝑖+ℎ2. 复合Gauss 积分的思想是将[𝑎,𝑏]作等距划分：𝑥𝑖=𝑎+𝑖ℎ(𝑖=0,…,𝑛), 然后在每个子区间内应用两点Gauss公式.试对步长ℎ作先验估计。（误差要求与(a)同）**

# 算法描述

1. 定义了两个辅助函数，计算1/x，1/(x\*x+1)，以及计算龙贝格公式所需要的项目。
2. Simpson算法

先计算出h值（通过传入的变量a，b，n）；

Sum=f(a)+f(b)+4\*f(a+h/2);

For(i从0到n-1)

【

Sum=sum+4\*f(a+h/2+k\*h/2)+2\*f(a+k\*h);  
】

输出结果

1. Romberg算法

仍然是先计算h值，b-a；

temp1[0]= (f(a)+f(b))\*h/2;

temp2[0]=temp1[0]/2+f(a,b,func,1),

temp2[1]=4/3\*temp2[0]-1.0/3\*temp1[0];

While (|t2[i-1]-t1[i-2]|> )

for j=1,2….,i-1

//外推

//更新temp1

//由temp1[n]计算temp2[n]

i=i+1

temp2[i-1]输出结果

1. Gauss算法

For k=0,2……,n-1

Sum+= + ;

Sum\*h/2

输出结果sum值

# 程序清单

（使用devc++）

#include<iostream>

#include<cmath>

using namespace std;

long double fuzhu\_1(double x,int t)

{

if(t==1)

return 1.0/x;

else

return 1.0/(1+x\*x);

}

long double fuzhu(double a,double b,int n,int t)

{

double h=(b-a)/n;

double sum=0.0;

for(int i=0;i<=n-1;i++)

{

sum=sum+fuzhu\_1(a+i\*h+h/2,t);

}

sum=sum\*(h/2);

return sum;

}

long double simpson(double a,double b,int n,int t)

{

double h=(b-a)/double(n);

double sum;

sum=fuzhu\_1(a,t)+fuzhu\_1(b,t)+4\*fuzhu\_1(a+h/2,t);//注意下标不同，不要搞错了

for(int i=1;i<=n-1;i++)

{

sum=sum+4\*fuzhu\_1(a+i\*h+h/2,t)+2\*fuzhu\_1(a+i\*h,t);

}

sum=sum\*(h/6);

return sum;

}//求取ln2,pai

long double romberg(double a,double b,int t)

{

double temp1[5000];

double temp2[5000];

double h=b-a;

int i=2;

temp1[0]=(fuzhu\_1(a,t)+fuzhu\_1(b,t))\*h/2;

temp2[0]=(temp1[0]/2)+fuzhu(a,b,1,t);

temp2[1]=pow(4.0,1)/(pow(4.0,1)-1)\*temp2[0]-1.0/(pow(4.0,1)-1)\*temp1[0];

while(abs(temp2[i-1]-temp1[i-2])>pow(10.0,-9))

{

//cout<<i<<" "<<temp2[i-1]<<" "<<temp1[i-2]<<endl;

for(int j=0;j<i;j++)

{

temp1[j]=temp2[j];

}

temp2[0]=temp1[0]/2+fuzhu(a,b,int(pow(2.0,i-1)),t);

for(int j=1;j<=i+1;j++)

{

temp2[j]=pow(4.0,j)/(pow(4.0,j)-1)\*temp2[j-1]-1.0/(pow(4.0,j)-1)\*temp1[j-1];

}

i++;

}

//cout<<"fuck"<<endl;

return temp2[i-1];

}

long double gauss(double a,double b,int n,int t)

{

double h=(b-a)/double(n);

double sum=0.0;

for(int i=0;i<=n-1;i++)

{

sum=sum+fuzhu\_1(a+i\*h+h/2-h/2/pow(3.0,0.5),t)+fuzhu\_1(a+i\*h+h/2+h/2/pow(3.0,0.5),t);

}

sum=sum\*h/2;

return sum;

}

int main()

{

double ln2=0.6931471806,pi=3.1415926535;

cout<<"simpson:"<<endl;

cout<<"ln2: "<<simpson(1,2,36,1)<<" 误差为："<<abs(simpson(1,2,36,1)-ln2)<<endl;

cout<<"pi: "<<4\*simpson(0,1,36,2)<<" 误差为："<<abs(4\*simpson(0,1,36,2)-pi)<<endl;

cout<<"romberg:"<<endl;

cout<<"ln2: "<<romberg(1,2,1)<<" 误差为 "<<abs(romberg(1,2,1)-ln2)<<endl;

cout<<"pi: "<<4\*romberg(0,1,2)<<" 误差为："<<abs(4\*romberg(0,1,2)-pi)<<endl;

cout<<"gauss:"<<endl;

cout<<"ln2: "<<gauss(1,2,33,1)<<" 误差为 "<<abs(gauss(1,2,33,1)-ln2)<<endl;

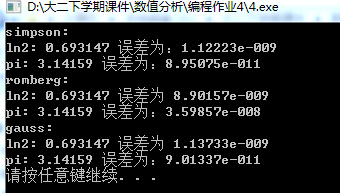
cout<<"pi: "<<4\*gauss(0,1,33,2)<<" 误差为："<<abs(4\*gauss(0,1,33,2)-pi)<<endl;

system("pause");

return 0;

}

# 体会与问题

1. 本次试验首先应当确认simpson公式和gauss公式中n的值，通过求导结合要求的误差限度simpson公式中n值为36，而gauss公式中n值应为33，通过笔算将n的值确定以后就可以使用计算机编程计算了。
2. 所得的结果如图所示，是符合计算要求的
3. 注意在编程的时候加入一个传入的参数可以避免重载，个人感觉这是我这个程序比较简洁的原因。
4. Romberg的运行起来n=6就终止了，相对于其他两种n值为30+的方法精确度提升很高，所以个人感觉这是一种提高精度极快的方法，以指数开始收敛，但是一旦n较大，运行起来也会比较慢。