# 实验三 数值积分(2 课时)

## 一、实验目的

1.了解数值积分的基本原理和方法。

2.掌握复合梯形公式。

3.了解求积公式外推思想、Romberg 公式及 Romberg 积分法。

## 二、实验要求

1.编写定步长复合梯形公式

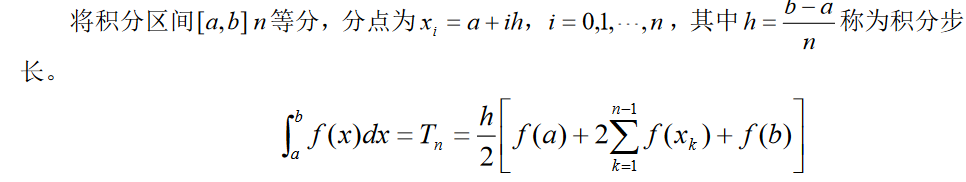
2.编写变步长复合梯形公式。

3.进一步加深对数值积分的理解。

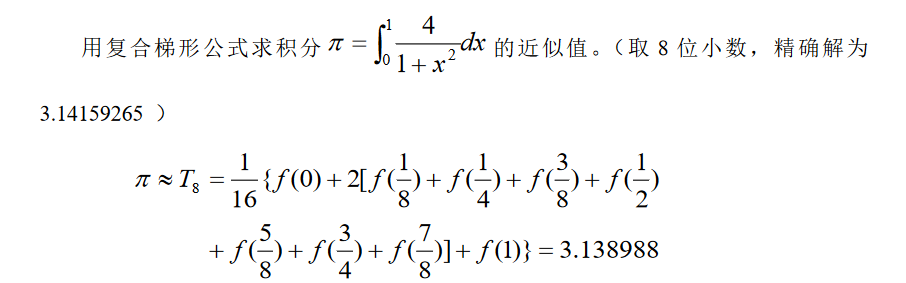
## 三、实验原理

**(一)定步长复合梯形公式**

1.公式

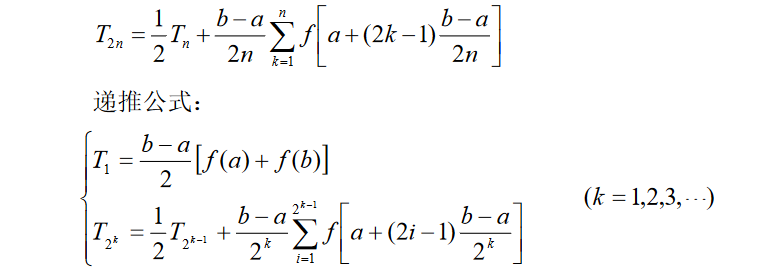


2. 例子

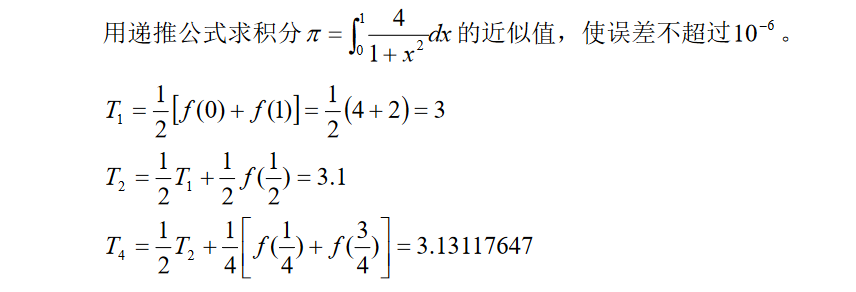


**(二) 变步长复合梯形公式**

1. 公式



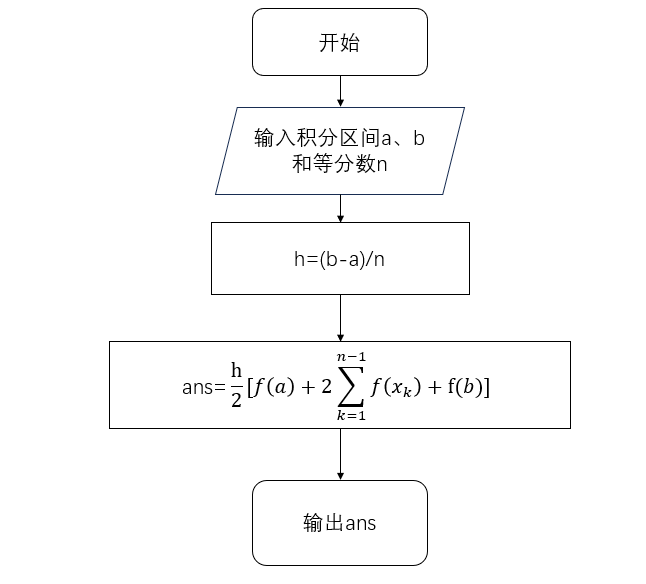
2. 例子



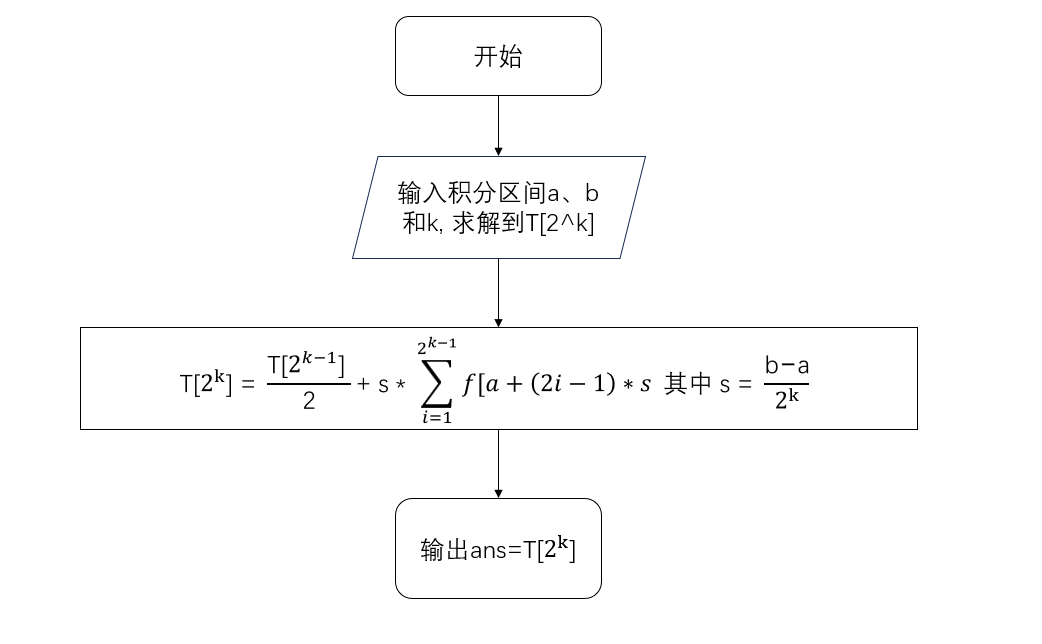
## 四、实验内容

**(一) 算法流程图**

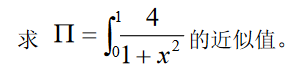
1. 定步长复合梯形算法流程图



2. 变步长复合梯形算法流程图



(二) 编程作业



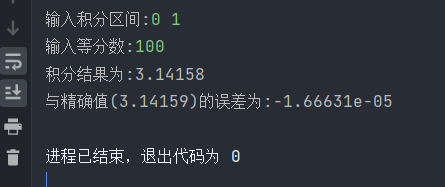
(1)编写定步长复合梯形程序求解上式;

(2)编写变步长复合梯形程序求解上式，使误差不超过 10-6.

【提示】请根据前面的算法流程图进行编写程序。

1.定步长复合梯形程序求解

*#include* <bits/stdc++.h>  
  
*using namespace* std;  
  
*double* f(*double* x) {  
 *return* 4 / (1 + x \* x);  
}  
  
*int* main() {  
 cout << "输入积分区间:";  
 *double* a, b;  
 *int* n;  
 cin >> a >> b;  
 cout << "输入等分数:";  
 cin >> n;  
 *double* h = (b - a) / n;  
 *double* ans = 0;  
 ans += f(a) + f(b);  
 *for* (*int* k = 1; k <= n - 1; k++) {  
 ans += 2 \* f(a + k \* h);  
 }  
 ans \*= h / 2;  
 cout << "积分结果为:" << ans << endl;  
 *double* pi = 3.14159265;  
 cout << "与精确值(" << pi << ")的误差为:" << ans - pi << endl;  
}



2.变步长复合梯形程序求解

*#include* <bits/stdc++.h>  
  
*using namespace* std;  
  
*double* f(*double* x) {  
 *return* 4 / (1 + x \* x);  
}  
  
*double* getTk1(*int* k, *double* a, *double* b) {  
 *double* T[(1 << k) + 1];  
 T[1] = (f(a) + f(b)) / 2;  
 *for* (*int* i = 2; i <= (1 << k); i <<= 1) {  
 *// T[2^k] = T[2^(k-1)]/2 + s \* sum{ f[a+(2i-1)\*s] | i=1->2^(k-1) } 其中 s = (b-a)/2^k* T[i] = T[i / 2] / 2;  
 *double* s = (b - a) / i;  
 *double* t = 0;*// t = sum{ f[a+(2i-1)\*s] | i=1->2^(k-1) }  
 for* (*int* j = 1; j <= i / 2; j++) {  
 *double* x = a + (2 \* j - 1) \* s;  
 t += f(x);  
 }  
 T[i] += s \* t;  
 }  
 *return* T[1 << k];  
}  
  
*double* getTk2(*int* k, *double* a, *double* b) {  
 *// 压缩T数组, 2^k -> k  
 double* T[k + 1];*// T[k] <--> T[2^k]* T[0] = (f(a) + f(b)) / 2;  
 *for* (*int* i = 1; i <= k; i++) {  
 *// T[2^k] = T[2^(k-1)]/2 + s \* sum{ f[a+(2i-1)\*s] | i=1->2^(k-1) } 其中 s = (b-a)/2^k  
 // T[i] = T[i-1]/2 + s \* sum{ f[a+(2j-1)\*s] | j=1->2^(i-1) } 其中 s = (b-a)/2^i* T[i] = T[i - 1] / 2;  
 *double* s = (b - a) / (1 << i);  
 *double* t = 0;*// t = sum{ f[a+(2i-1)\*s] | i=1->2^(k-1) }  
 for* (*int* j = 1; j <= (1 << (i - 1)); j++) {  
 *double* x = a + (2 \* j - 1) \* s;  
 t += f(x);  
 }  
 T[i] += s \* t;  
 }  
 *return* T[k];  
}  
  
*double* getTk3(*int* k, *double* a, *double* b) {  
 *// T的后一项只依靠前一项,所以仅使用一个变量存储即可  
 double* T = (f(a) + f(b)) / 2;  
 *for* (*int* i = 1; i <= k; i++) {  
 *// T[2^k] = T[2^(k-1)]/2 + s \* sum{ f[a+(2i-1)\*s] | i=1->2^(k-1) } 其中 s = (b-a)/2^k  
 // T[i] = T[i-1]/2 + s \* sum{ f[a+(2j-1)\*s] | j=1->2^(i-1) } 其中 s = (b-a)/2^i* T /= 2;  
 *double* s = (b - a) / (1 << i);  
 *double* t = 0;*// t = sum{ f[a+(2i-1)\*s] | i=1->2^(k-1) }  
 for* (*int* j = 1; j <= (1 << (i - 1)); j++) {  
 *double* x = a + (2 \* j - 1) \* s;  
 t += f(x);  
 }  
 T += s \* t;  
 }  
 *return* T;  
}  
  
*int* main() {  
 cout << "输入积分区间:";  
 *double* a, b;  
 *int* k;  
 cin >> a >> b;  
 cout << "输入k(求到T[2^k]):";  
 cin >> k;  
  
 *double* ans = getTk3(k, a, b);  
 cout << "积分结果为:" << ans << endl;  
 *double* pi = 3.14159265;  
 cout << "与精确值(" << pi << ")的误差为:" << ans - pi << endl;  
}

