《计算方法》上机作业报告

学生信息：

|  |  |  |
| --- | --- | --- |
| 姓名：何宇昊 | 学号：3190102182 | 授课教师：郑太英 |

## 上机作业内容：

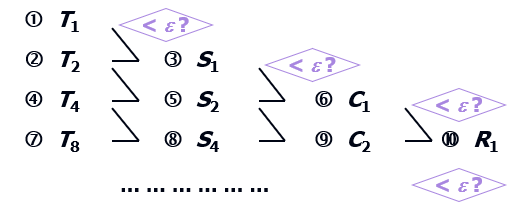
**龙贝格数值积分方法**

## 算法目的和意义

当被积函数f(x)在区间［a,b］上连续时,要使得复合梯形公式比较精确地代替定积分，可将分点(即基点)加密,也就是将区间［a,b］细分,然后利用复合梯形公式求积。而实际计算过程中，梯形公式积分值较为粗糙。而我们可以在步长逐步分半过程中将粗糙的积分值逐步加工为精度较高的积分值.

将收敛缓慢的梯形值序列加工成收敛迅速的积分值序列，这种加速方法称为龙贝格算法。

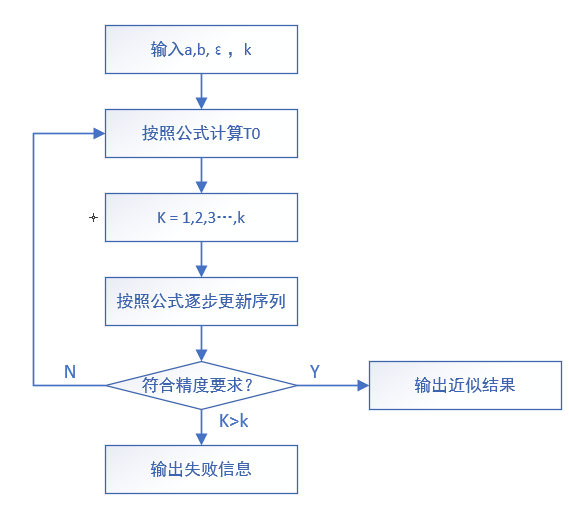
## 计算公式



## 程序设计

这个程序设计过程中需要注意的地方有：

1. 在计算出龙贝格序列第一项之前，如果用矩阵来存储积分值，那么是存储在一个下三角矩阵中
2. 在和精度进行比较的时候，每个序列一定要有两个值以上才能开始比较，并且只要比较符合精度范围就退出，无论其是否处于龙贝格序列。
3. 矩阵一列存储一个数列，每计算完一行就新加一行，矩阵大小随着迭代变化。
4. 注意梯形公式变步长的迭代公式，每个新出现的函数点之间的距离是上一轮迭代的时候的步长相等，
5. 程序根据计算梯形公式序列，计算辛普生序列，计算科特斯序列，计算龙贝格序列这样反复进行，直到符合精度要求退出

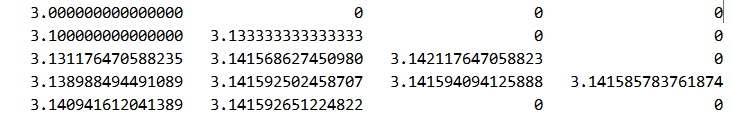
程序框图  


## 结果讨论与分析

对上述积分在截断误差不大于1e-6的条件下进行近似计算

计算得到的结果是：3.141592651224822

得到的按照一列一序列排布是：



可以看到其较为灵敏地在最后一次计算Simpson公式的时候就停下并返回，灵敏度较高。

**程序评价：**

这个程序较好完成了任务，除了每次迭代的时候需要更改矩阵容量稍微显得有点拖沓之外，整个程序力图使自己看起来更加简洁。同时，程序的适应性也更好。

亮点：

1. 用户可以自己输入需要拟合地函数，而不需要进入函数内部修改
2. 用二维数组存储四条序列，迭代过程巧妙应用了列号和迭代公式之间的关系
3. 用了比较巧妙的方法收集梯形公式中新出现地函数点，并且利用函数句柄一次性计算完。

## Appendix ( coding in matlab)

【Lab3.m】

clc;

clear;

% 添加参数

bound = input('请根据[a,b]输入积分上下限，a为下限，b为上限\n');

epsilon = input('请输入精度要求epsilon \n');

k = input('请输入最大二分次数k \n');

s=input('请输入函数表达式(默认变量为x)：y = \n','s');

y = matlabFunction(str2sym(s));

result = Romberg(y,bound,epsilon,k);

% 输出结果

disp(result);

【Romberg函数本身】

function result = Romberg(y,bound,epsilon,k)

    new\_rows = zeros(1,4);

    Sequence = new\_rows;

    Sequence(1,1) = sum(y(bound))/2;

    i = 1;

    step\_size = bound(2)-bound(1);%初始化步长

    while i<=k

        for j = 2:min(i,4)%因为龙贝格算法只考虑到龙贝格序列，因此只有四列

            Sequence(i,j) = (4^(j-1)\*Sequence(i,j-1)-Sequence(i-1,j-1))/(4^(j-1)-1);

            if i>j && abs(Sequence(i,j)-Sequence(i-1,j)) < epsilon

                %这里是避免序列中的值非常接近0，所以避免在每个序列地第一项就进行比较

                %满足条件就退出

                result = Sequence(i,j);

                disp(Sequence);%展示龙贝格序列，测试代码可删去

                return

            end

        end

        %增多新的一行，本可以预处理足够大的空间，但觉得没必要

        Sequence = [Sequence;new\_rows];

        i = i+1;

        %更新新的梯形序列

        [Sequence(i,1),step\_size] = StepSizeVaried(y,bound,step\_size,Sequence(i-1,1));

    end

    result = -1;

    disp("Fail");

end

【变步长求梯形公式】

function [result,new\_stepsize] = StepSizeVaried(y,bound,last\_stepsize,last\_value)

% 这个函数就是用来更新T\_n 到T\_2n，返回新步长

    new\_stepsize = last\_stepsize/2;

    new\_x = bound(1)+new\_stepsize : last\_stepsize : bound(2);%创建新的多出来的序列

    new\_y = y(new\_x);%使用句柄计算新序列的函数值

    result = last\_value/2 + new\_stepsize\*sum(new\_y);%更新

end