第14周作业

——2023210314，赵熠卓

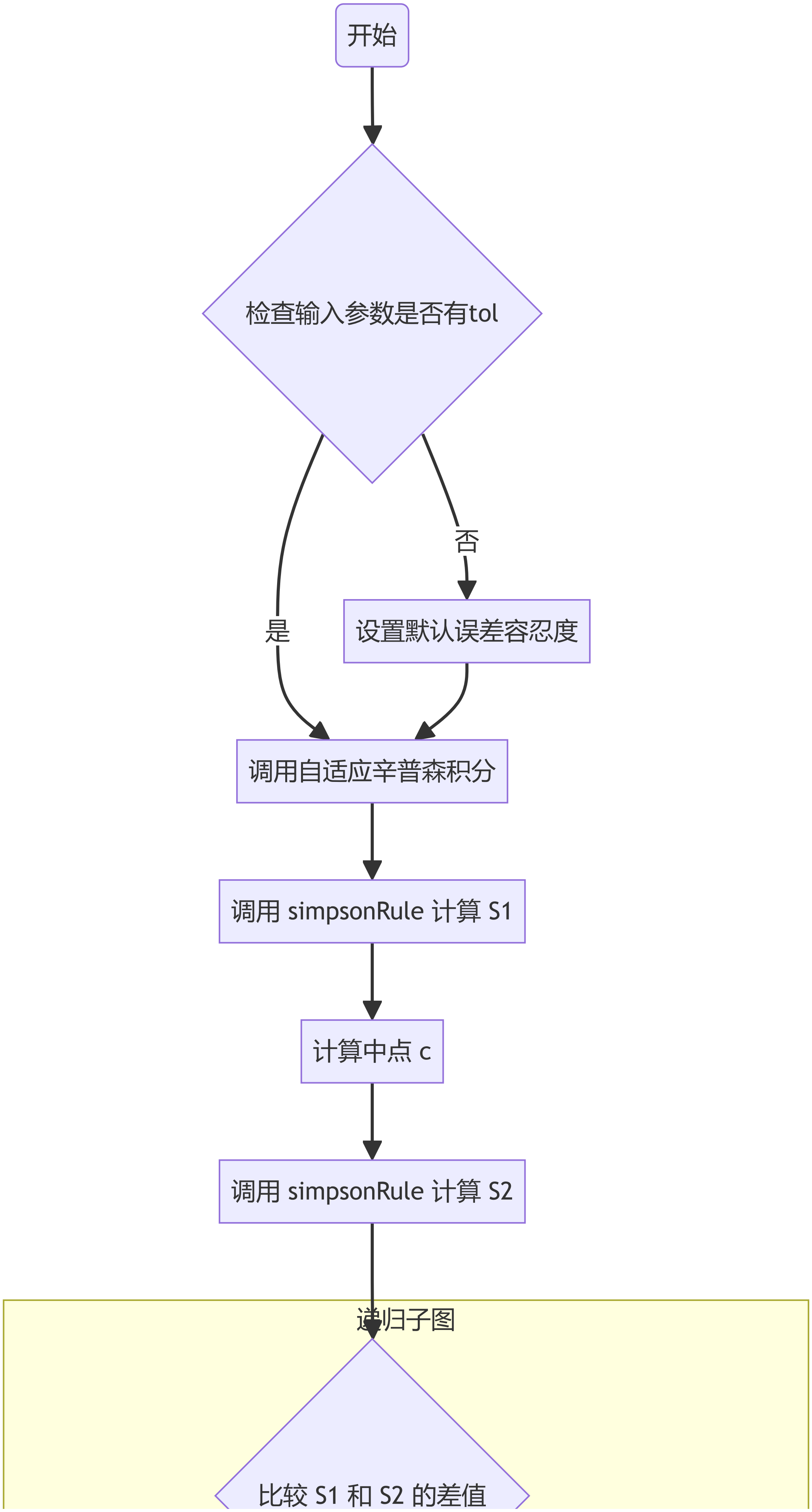
Question1

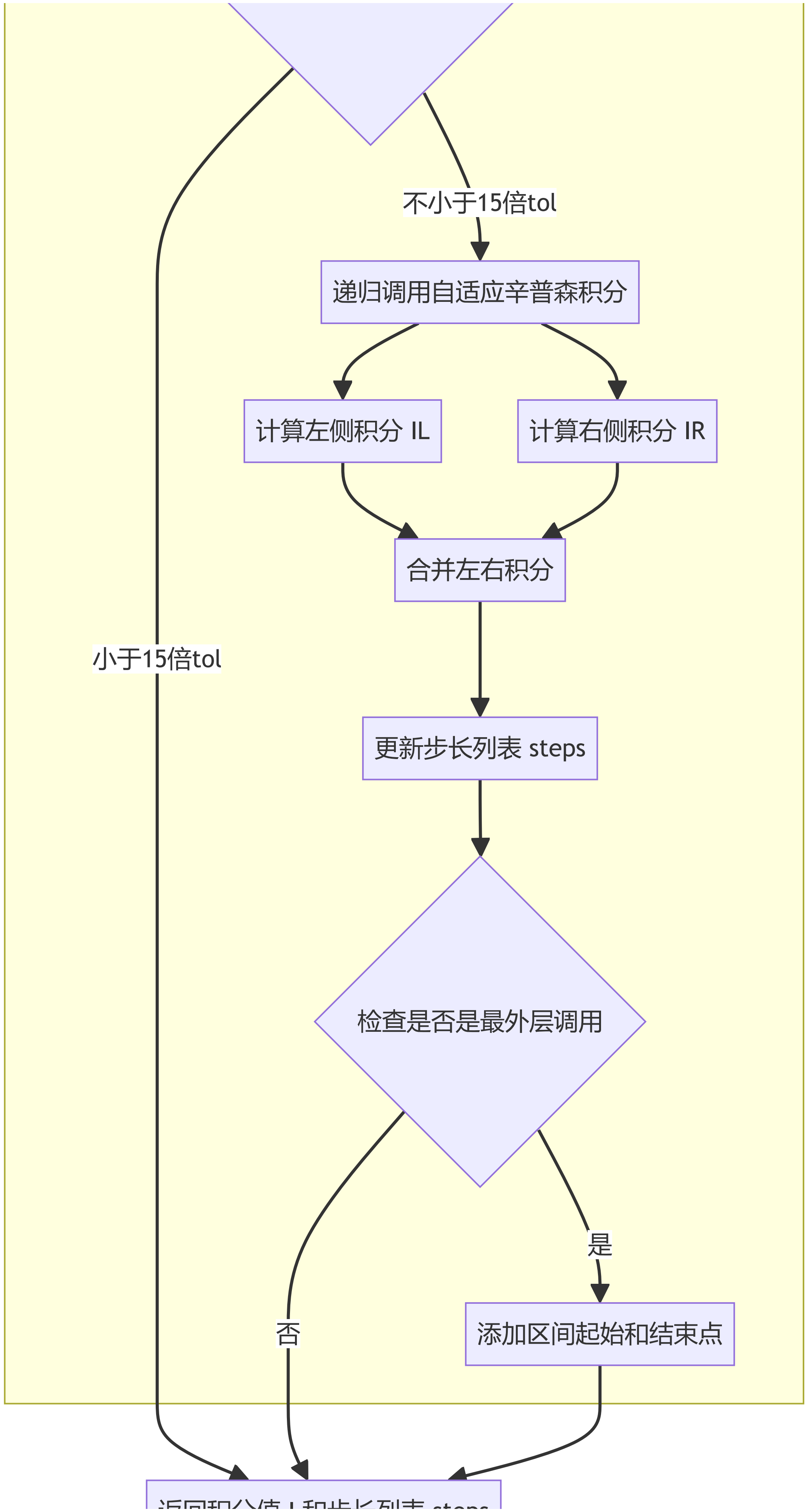
(一).问题描述

编写复化自适应步长辛普森积分函数，实现对函数$f(x) = \frac{sin(x)}{x}$在$[1, 5]$区间的积分，并分析各分段积分区间的自适应步长值。

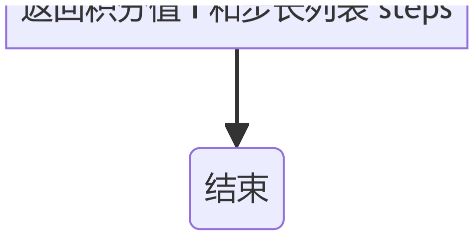
(二).程序设计

A.流程图


开始
检查输入参数是否有tol
是
否
设置默认误差容忍度
调用自适应辛普森积分
调用 simpsonRule 计算 S1
计算中点 c
调用 simpsonRule 计算 S2
递归子图
比较 S1 和 S2 的差值



不小于15倍tol
递归调用自适应辛普森积分
计算左侧积分 IL
计算右侧积分 IR
合并左右积分
小于15倍tol
更新步长列表 steps
检查是否是最外层调用
是
否
添加区间起始和结束点


返回积分值 I 和步长列表 steps
结束

B.代码

```matlab
%simpson.m
function [I, steps] = simpson(fun, a, b, tol, varargin)
% 复化自适应步长辛普森公式求解数值积分，并返回区间划分步长
% 输入参数:
% fun: 被积函数
% a, b: 积分区间的端点
% tol: 误差容忍度
% varargin: 函数fun的附加参数
% 输出参数:
% I: 求得的积分值
% steps: 区间划分的步长列表

  if nargin < 4 || isempty(tol)
    tol = 1e-4; % 默认误差容忍度
  end

  [I, steps] = adaptiveSimpson(fun, a, b, tol, a, b, varargin{:});
end

function [I, steps] = adaptiveSimpson(fun, a, b, tol, fullA, fullB, varargin)
  % 自适应辛普森积分辅助函数
  % 使用递归方法自动调整积分步长以满足误差要求，并记录每次划分的中点

  c = (a + b) / 2;
  S1 = simpsonRule(fun, a, b, varargin{:});
  S2 = simpsonRule(fun, a, c, varargin{:}) + simpsonRule(fun, c, b, varargin{:});

  if abs(S1 - S2) < 15 * tol
    I = S2 + (S2 - S1) / 15;
    steps = c;
  else
    [IL, stepsL] = adaptiveSimpson(fun, a, c, tol/2, fullA, fullB, varargin{:});
    [IR, stepsR] = adaptiveSimpson(fun, c, b, tol/2, fullA, fullB, varargin{:});
    I = IL + IR;
    steps = [stepsL, c, stepsR];
  end

  % 如果是最外层调用，则添加区间的起始和结束点
  if a == fullA && b == fullB
    steps = [a, steps, b];
  end
```

```
end

function S = simpsonRule(fun, a, b, varargin)
  % 辛普森规则计算函数
  c = (a + b) / 2;
  fa = feval(fun, a, varargin{:});
  fb = feval(fun, b, varargin{:});
  fc = feval(fun, c, varargin{:});
  S = (b - a) * (fa + 4*fc + fb) / 6;
end
```

(三).计算结果与分析

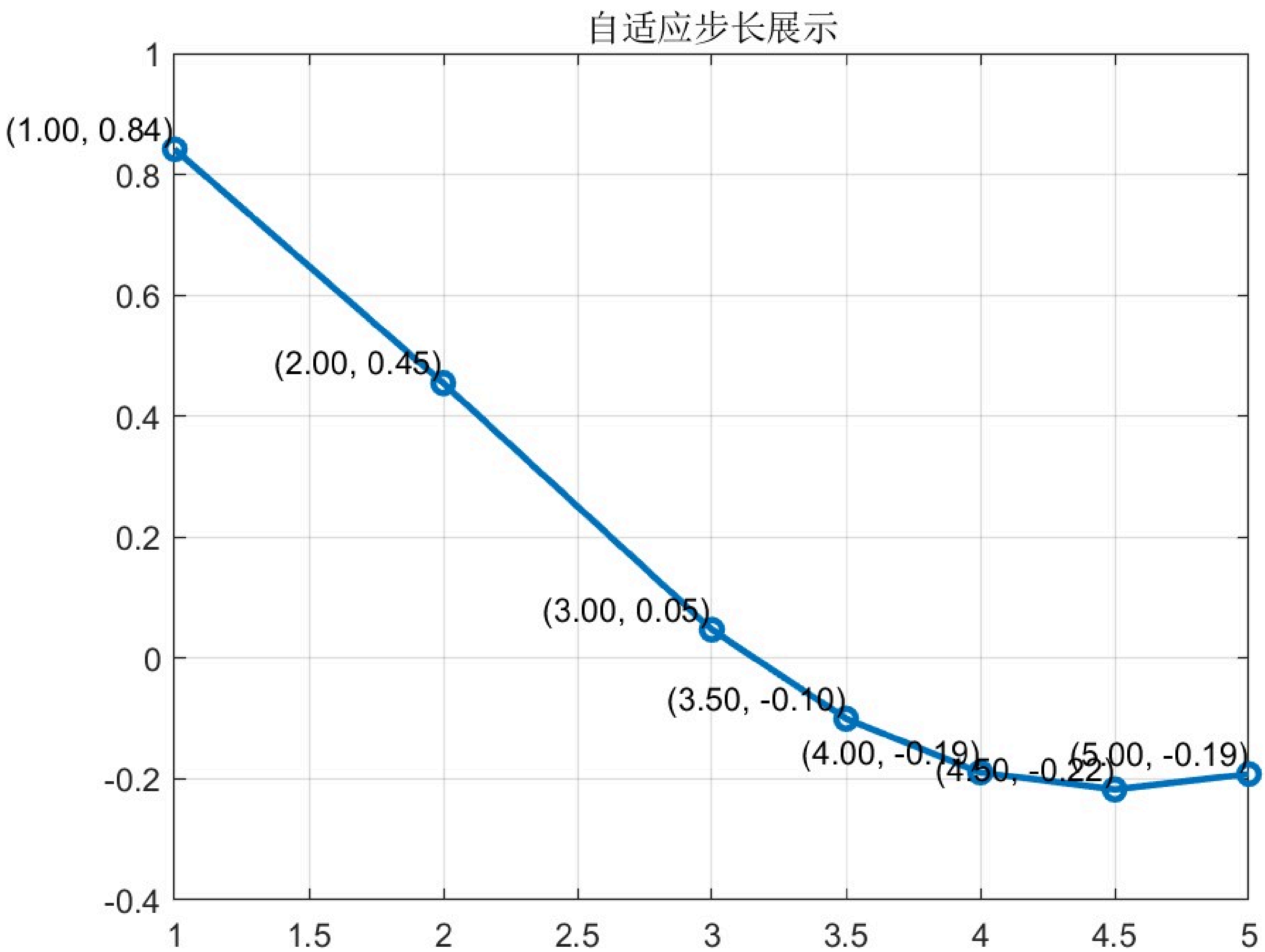
代码

```
%test1.m
% 定义函数
fun = @(x) sin(x)./x;

% 求积分，例如从 0 到 10，误差容忍度为 1e-6
[I, steps] = simpson(fun, 1, 5, 1e-6);

% 显示结果
fprintf('积分结果为: %f\n', I);

y_values = arrayfun(fun, steps);

% 绘制步长点
plot(steps, y_values, 'o-', 'LineWidth', 2);
title('自适应步长展示');
xlabel('时间 (秒)');
ylabel('速度 (m/s)');
grid on;

% 增加注释
for i = 1:length(steps)
    text(steps(i), y_values(i), sprintf('(%0.2f, %0.2f)', steps(i), y_values(i)), ...
         'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right');
end
```

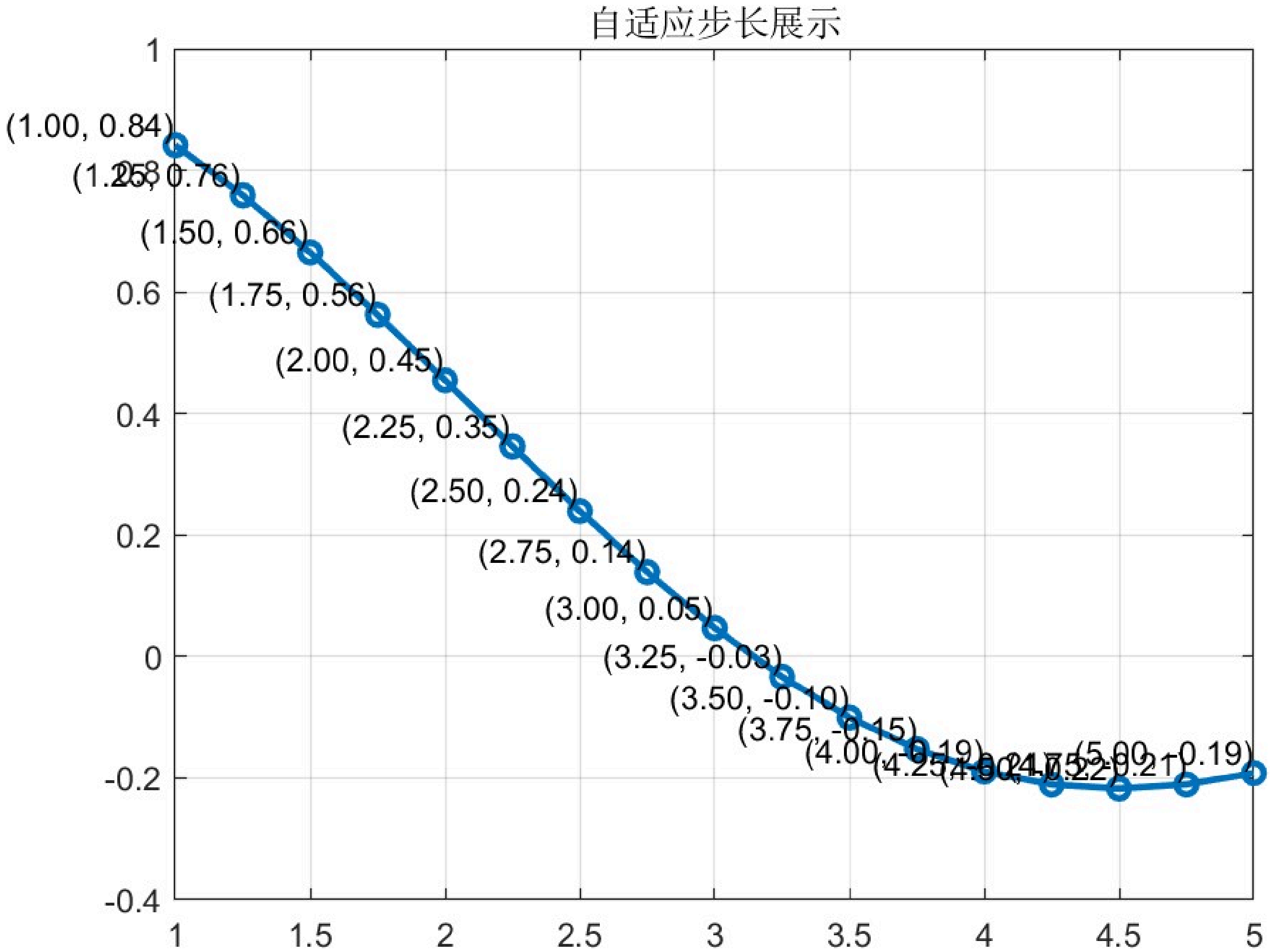
```
>> test1
积分结果为: 0.603848
```

分析

精度为$1e^{-4}$时图像如下:



精度为$1e^{-6}$时图像如下:



在函数二阶导的绝对值比较小的时候，区间也比较稀疏，在二阶导的绝对值比较大的时候，区间更密集；对于更高的精度要求，也会相应的分出更密的区间。