上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

上海交通大学 物理与天文学院
SJTU School of Physics and Astronomy

# Simpson 法数值积分

## 分析与优化


李政宏

2025.5.9

第一部分

# 计算结果与分析

数值计算与初步分析

# 问题回顾与数值计算

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

## 内容

- 计算结果
- 换用不同步长，记录时间与误差
- 误差分析与作图

# 计算结果

目标函数为：

$$\int_0^1 \frac{x^4 \cdot (1-x)^4}{1+x^2} (= \frac{22}{7} - \pi) \tag{1}$$

采用辛普森法进行计算，原理如下：

$$\int_a^b f(x)dx \approx \frac{h}{3}(f(a) + 4f(\frac{a+b}{2}) + f(b)) \tag{2}$$

其中 $h = \frac{b-a}{n}$ 为步长，$n$ 为分割数。

# 计算结果

**Python 代码 (步长取 1e-8)**

```python
starttime = time.time()
for i in range(n):
    x1=down+i*h
    x2=down+(i+1)*h
    result+=f(x1)+f(x2)+4*f((x1+x2)/2) # 计算每个小区间的函数值
result = result * h / 6 # 计算积分值
endtime = time.time()
usetime = endtime - starttime
error=abs(result-22/7+pi) # 计算误差
```

输出：积分结果为：0.001264489267349599 误差为：0.0(超出存储精度) 计算时间为：1.043898344039917 秒

# 不同步长重复计算

Python 代码

```python
different_h =np.linspace(1, 1e-3, 1000)
'''Initializing the result, error, time lists: np.zeros(1000)'''
for i in range(1000):
    h = different_h[i]
    starttime = time.time()
    '''the same calculation as previous page'''
    endtime = time.time()
    '''call append() for time, error, result'''
```

# 计算时间，计算结果，计算误差与步长的关系

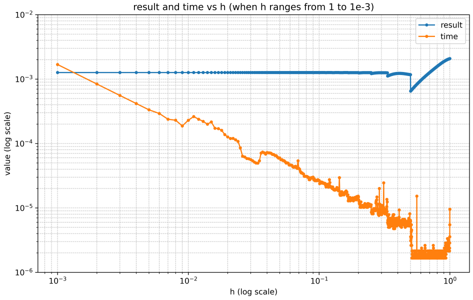


**图:** 计算时间, 计算结果与步长的关系

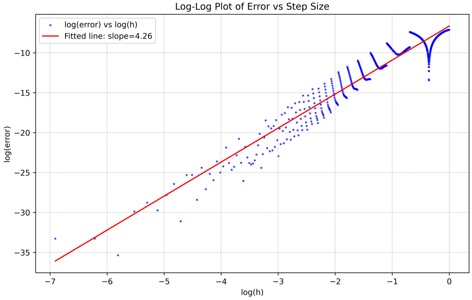


**图:** 计算误差与步长的关系

# 误差分析

## 宏观趋势

当步长小于 $1 \times 10^{-3}$ 后，误差精度过小，超出默认的浮点数存储精度，故本代码的分析中 $h$ 下限为 $1 \times 10^{-3}$。
在可分辨的范围内，误差对数和步长对数大体上为线性关系，但会在一小段区间内先**下降再突变上升**。
使用 `numpy` 库的 `polyfit` 线性拟合，得到斜率大体为 4.26，即 $e \propto h^{4.26}$ 根据课堂讲解得知

$$e \propto h^4$$

故应当写成

$$e = 10^{0.26} \cdot h^4 \approx 1.82 \cdot h^4$$

# 误差分析-震荡上升

## 局部截断误差与全局累积误差

- **局部截断误差**：
  辛普森法在单个子区间 $[x_i, x_{i+1}]$ 上的误差由泰勒展开的高阶项主导，理论公式为：
  $$e_{\text{local}} \propto h^5 \cdot \max\left|f^{(4)}(x)\right|$$
  即局部误差与 $h^5$ 和函数四阶导数的幅值成正比。
- **全局累积误差**：
  全局误差是各子区间误差的叠加。则全局误差满足：
  $$e_{\text{global}} = N \cdot e_{\text{local}} \propto \frac{L}{h} \cdot h^5 = L \cdot h^4$$

# 误差分析-震荡上升

**误差的震荡现象源于两种机制的竞争：**

1. **截断误差主导区（$h$ 较大时）**：
   - 步长 $h$ 减小时，局部截断误差 $\propto h^5$ 快速下降，全局误差随之按 $h^4$ 标度减小。
   - 但当 $h$ 过小时，子区间数 $N \propto 1/h$ 急剧增加，导致浮点运算的**舍入误差**通过 $N$ 次累积被放大。
2. **舍入误差主导区（$h$ 过小时）**：
   - 当 $h$ 小于临界值 $h_c$ 时，舍入误差 $\propto N \cdot \epsilon_{\text{machine}} \approx \epsilon_{\text{machine}}/h$ 开始主导，整体误差随 $h$ 减小而**反向增大**。
   - 临界步长 $h_c$ 由截断误差与舍入误差相等确定，典型值约为 $h_c \sim \epsilon_{\text{machine}}^{1/5}$（对双精度浮点数 $h_c \sim 10^{-3}$）。

这一竞争机制导致误差曲线在 $h_c$ 附近出现极小值，并在 $h < h_c$ 时发生突升，形成非单调震荡现象。

第二部分

# 优化与提升

自适应优化

# 自适应辛普森积分优化

当我们对时间的要求高而对精度要求变低时，传统辛普森积分法在计算速度上存在缺陷 (事实上，任何固定步长的积分都避免不了 $O(n)$ 的缺陷)，我们可以通过自适应积分进行优化–在积分过程中不断调整步长.

# 自适应积分的原理

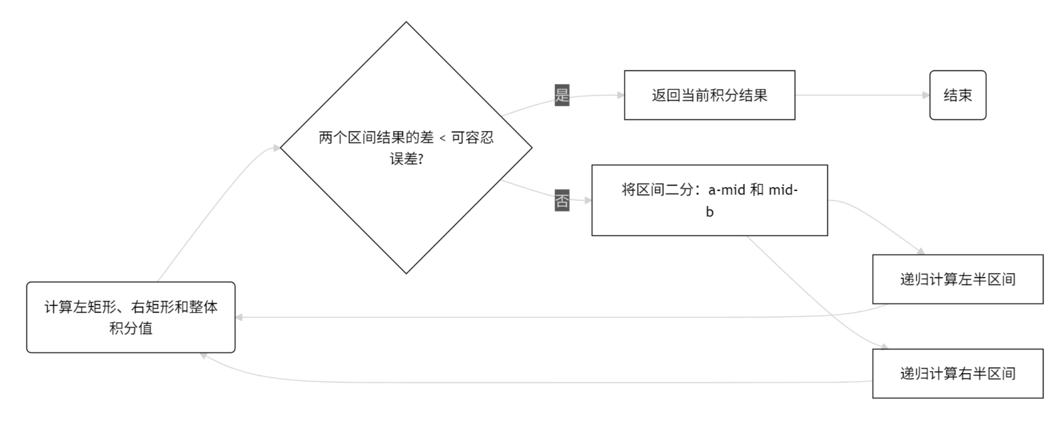


**图:** 自适应积分的流程图

# 代码实现

## 自适应积分

```
def recur_int(func, a, b, tol):
    mid = (a + b) / 2
    left = (b - a) * (func(a) + 4 * func((a + b) / 2) + func(b)) / 6
    right = ((mid - a) * (func(a) + 4 * func((a + mid) / 2) +
    func(mid)) / 6 +
    (b - mid) * (func(mid) + 4 * func((mid + b) / 2) + func(b)) / 6)
    if abs(left - right) < tol: # 误差小于容忍度，返回右侧积分值
        return right
    else:
        return recur_int(func, a, mid, tol / 2) +
        recur_int(func, mid, b, tol / 2) # 误差较大，递归调用，继续细分区间
    return recur_int(func, a, b, tol)
```

# 自适应积分的结果

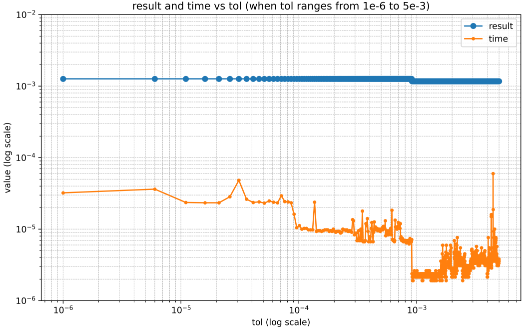


**图:** 自适应积分的结果

- 可以看到在 1e-3 的容差之内，保持高精度的结果下运算时间 (由于电脑的性能不稳定，每次运算结果可能在细节上有差异) 缩短了一个数量级。
- 在积分区间很大且存在平缓区域时，自适应积分性能优异，不过这归根到底是空间换时间的做法，当然，也可以通过优化代码实现不递归的写法。

# 结论

- Simpson 积分法在对本函数积分时，$0 \sim 1$ 区间内误差满足 $O(h^4)$，系数为 1.82。
- 在一个小区间内，浮点数精度的舍入误差和函数的近似误差先后占据主导。误差整体上随步长增加而增加，在小区域内随步长增加而降低。
- 通过自适应积分法，可以在保留 $1 \times 10^{-6}$ 精度下将运算速度提升一个数量级。

如果觉得代码不错，可以求个星星吗:)
https://github.com/k1-star/computational-physics

# Thanks

## 李政宏 · Simpson 法数值积分