分类号 _____ 编号_____

U D C_____ 密级_____

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

本科生毕业设计（论文）

题　　目： 计算 10 重积分

姓　　名： 王昊阳

学　　号： 12210202

系　　别： 物理系

专　　业： 物理学

指导教师： 黄丽 教授

2025 年 3 月 22 日

诚信承诺书

1. 本人郑重承诺所呈交的毕业设计（论文），是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。

2. 除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。

3. 本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4. 在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

作者签名:

_____ 年____ 月____ 日

目录

1. 问题阐述与分析

重积分是一种重要的多元积分，其被积函数通常具有不止一个变量，积分范围也不再是一个区间，而是多维空间中的一部分。在计算重积分的过程中，其积分难度通常会随着重数变多而增大。在维数不低于 4 的情况下，蒙特卡洛方法积分比梯形法，辛普森方法这一类的数值积分方法更具有优势。

本文将会用蒙特卡洛方法计算如下的 10 重积分并分析其结果，

$$I = \int_0^1 \cdots \int_0^1 \int_0^1 (x_1 + x_2 + \cdots + x_{10})^2 \, \mathrm{d}x_1 \, \mathrm{d}x_2 \cdots \mathrm{d}x_{10} \tag{1}$$

其解析结果为 $155/6$。

2. 问题中涉及到的变量

- 被积函数的变量组成的一个 10 维数组 $x = [x_1, x_2, \cdots, x_{10}]$
- 蒙特卡洛积分中的样本数量 N，需要是整数
- 蒙特卡洛积分的标准误差 `standard_error`
- 蒙特卡洛积分的相对误差 `relative_error`
- 蒙特卡洛误差 `montecar_error`

3. 计算模拟原理

本程序使用蒙特卡罗方法对被积函数进行数值积分，并会计算每一次蒙特卡洛积分的标准误差和积分结果与真实值的相对误差。

3.1 蒙特卡洛积分

蒙特卡洛积分的原理是在被积空间中随机均匀撒点，再求出这些点对应的被积函数的函数值并做平均处理，平均值乘以被积空间的体积即为蒙特卡洛方法的积分估计值。

在本题中，被积函数具有 10 个变量，每个变量的积分区间为 $[0, 1]$，因此在 10 维

空间中总的被积空间的体积 $V=1$。若撒出的点的个数为 N，蒙特卡洛积分公式为：

$$I=\frac{V}{N}\sum_{i}^{N}f(x_{1_i},x_{2_i},\cdots,x_{10_i}) \tag{2}$$

其中 $f(x_1,x_2,\cdots,x_{10})=(x_1+x_2+\cdots+x_{10})^2$ 是被积函数。

3.2 标准误差和蒙特卡洛误差的计算

由于每次蒙特卡洛积分会撒出 N 个点，因此会获得 N 个样本，从而可以进一步计算求得的积分的标准误差，即：

$$\sigma_f=\sqrt{\langle f^2\rangle-\langle f\rangle^2} \tag{3}$$

其中 $\langle f^2\rangle=\frac{1}{N}\sum f^2$，$\langle f\rangle=\frac{1}{N}\sum f$，进而可以进一步得到蒙特卡洛积分的一些稳定性分析和统计分析。

由于蒙特卡洛方法得到的积分值的精度会随着样本数量的增加以 $1/\sqrt{N}$ 的形式提高，通常会定义一个蒙特卡洛误差判断其精度。

$$\epsilon=V\frac{\sigma_f}{\sqrt{N}} \tag{4}$$

3.3 相对误差的计算

蒙特卡洛积分得到的积分值只是一个数值上的估计值，对于该积分，有解析求得的真实值 $I_{real}=\frac{155}{6}$。因此可以计算出蒙特卡洛积分的相对误差：

$$\delta=\frac{|I-I_{real}|}{I_{real}} \tag{5}$$

其中 I 是蒙特卡洛方法得到的积分估计值。

4. 程序设计

程序的流程图如下所示（图1），图中展示了程序的主流程以及程序中定义的两个函数 `mc_integral` 和 `error_array` 的子流程：

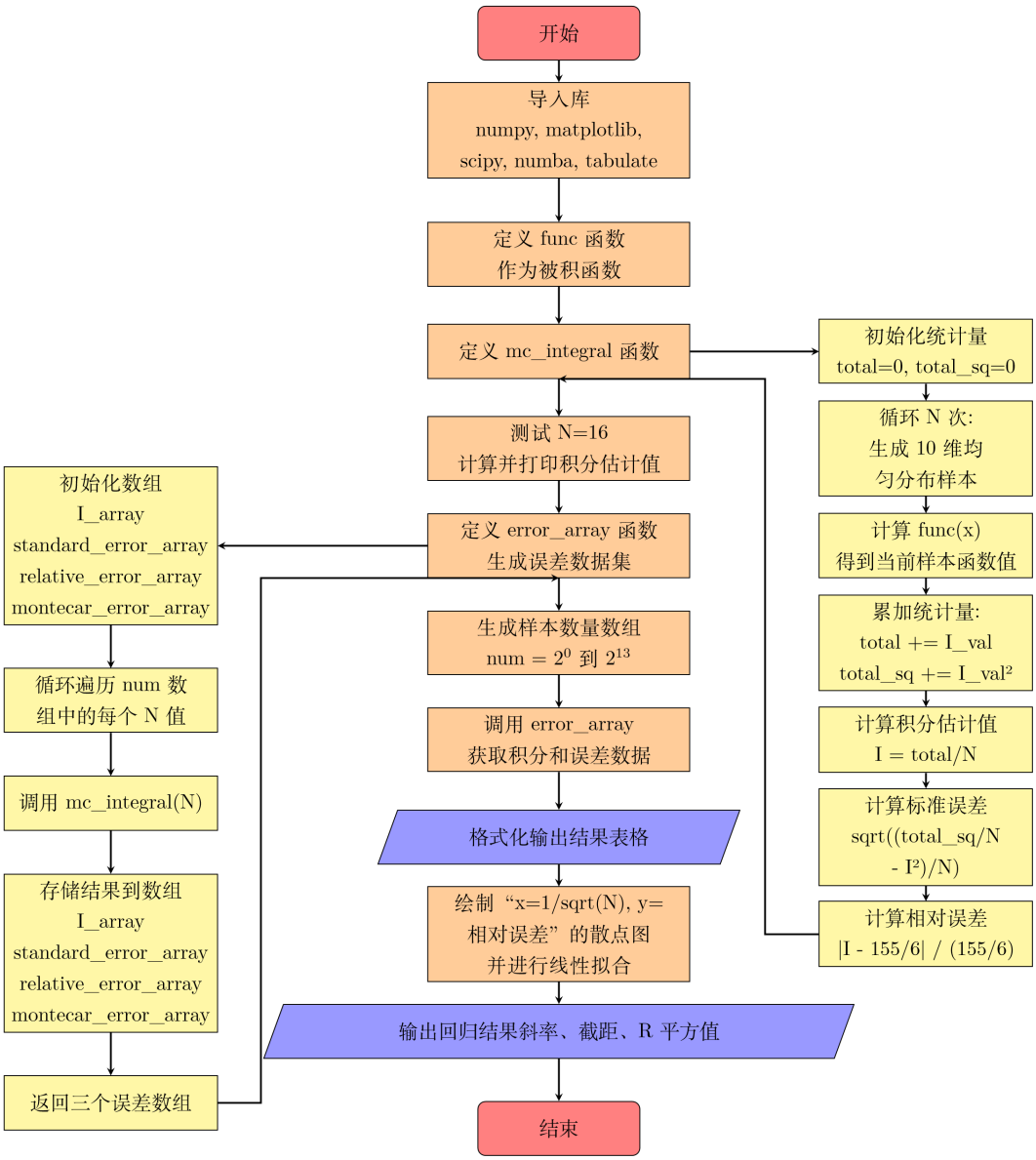

图 1 程序流程图

4.1 库的调用与被积函数的定义

程序需要使用 `numpy` 库便于随机数据的生成，使用 `matplotlib` 库进行绘图。`numba` 库用于加速 Python 函数的调用和计算，`scipy` 库用于统计分析和线性回归，`tabulate` 库则用于生成格式化的表格。

```
# 导入必要的库
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats  # 用于统计分析和线性回归
```

```
from numba import jit  # 用于加速函数计算
from tabulate import tabulate  # 用于生成格式化的表格
```

接下来定义被积函数 $f(x_1, x_2, \cdots, x_{10}) = (x_1 + x_2 + \cdots + x_{10})^2$。在程序中，被积函数被定义为`func(x)`函数，其变量`x`是一个数组，其函数值为数组`x`的所有元素的加和再平方。

```
# 使用 numba 加速的函数，计算 10 维向量 x 各元素和的平方
@jit(nopython=True)
def func(x):
    s = np.sum(x)
    return s * s
```

4.2 定义蒙特卡洛积分函数 mc_integral(N)

下面是用蒙特卡洛方法计算积分 I 的主要步骤。通过生成一个长度为 10，元素为 0 到 1 之间的均匀随机数的数组`x`作为被积函数`func(x)`的宗量，可以计算出相应的函数值。重复 N 次此过程再根据式2计算出蒙特卡洛积分估计值，再根据式3和式5分别计算出本次蒙特卡洛积分的标准误差和相对误差，便于下一步的统计和分析。

```
# 蒙特卡洛积分计算函数，N 为样本数量
@jit(nopython=True)
def mc_integral(N):
    total = 0  # 存储积分值的累加和
    total_sq = 0  # 存储积分值平方的累加和
    true_value = 155.0 / 6.0  # 理论精确值，计算 10 维超立方体上 (sum
    ↪  x_i)^2 的积分
    for _ in range(N):
        x = np.random.uniform(0, 1, 10)  # 生成 10 维均匀分布随机样
        ↪  本
        I_val = func(x)  # 计算当前样本的函数值
        total += I_val  # 累加函数值
        total_sq += I_val * I_val  # 累加函数值的平方
    I = total / N  # 计算蒙特卡洛积分估计值
```

```
    # 计算标准误差：sqrt(Var/N)，其中 Var 是样本方差
    standard_error = np.sqrt((total_sq / N - I * I) / N)
    # 计算相对误差：| 估计值 - 真实值 | / 真实值
    relative_error = np.abs(I - true_value) / true_value
    return I, standard_error, relative_error
```

对于 $N = 16$ 的情况，则可以直接通过`mc_integral(N)`函数计算出对应的积分值，代码如下：

```
# 测试 N=16 时的积分估计值
I = mc_integral(16)[0]
print(I)
```

得到的积分结果为：

```
26.577936994048187
```

4.3 定义 error_array(num) 函数用于生成各类误差的数组

为了探究蒙特卡洛积分估计值以及各类误差与样本数量 N 的关系，最好将不同的样本数对应的积分估计值及各类误差各自写成一个数组的形式，本程序中就通过定义`error_array(num)`函数来生成这样的数组。与函数关联的变量`num`是一个数组，它的每一个元素代表了该次蒙特卡洛积分的样本数量。在对该数组的各个元素循环的过程中，会用`mc_integral(N)`函数计算相应的积分估计值和各类误差，最终这些数据被写入四个数组中，`I_array`中存储积分估计值，`standard_error_array`中存储标准误差，`relative_error_array`中存储相对误差，`montecar_error_array`中存储蒙特卡洛误差。

```
# 生成不同样本量下的误差数据
@jit(nopython=True)
def error_array(num):
```

```
    I_array = []  # 存储不同 N 对应的积分估计值
    standard_error_array = []  # 存储不同 N 对应的标准误差
    relative_error_array = []  # 存储不同 N 对应的相对误差
    montecar_error_array = [] # 存储不同 N 对应的蒙特卡洛误差
    for i in num:
        # 调用蒙特卡洛积分函数获取结果
        I, standard_error, relative_error = mc_integral(i)
        I_array.append(I)
        standard_error_array.append(standard_error)
        relative_error_array.append(relative_error)
        montecar_error_array.append(standard_error / np.sqrt(i))
    return I_array, standard_error_array, relative_error_array,
    ↪  montecar_error_array
```

对于一个具体的例子，`num=1,2,4,8,⋯,8192` 即样本数量从 2^0 个以 2 为底用等比数列的形式一直取到 2^{13}，数组长度为 14，就可以使用以上的函数计算出与样本数量相对应的积分估计值和各类误差，并以表格的形式结构化输出这些数据。

```
# 生成样本数量数组：2^0, 2^1, ..., 2^13, 共 14 个点
num = np.logspace(0, 13, num=14, base=2, dtype=int)

# 获取不同样本量下的积分结果和误差
I_array, standard_error_array, relative_error_array,
↪  montecar_error_array = error_array(num)

# 使用 tabulate 库格式化输出结果表格
headers = ["N", "Integral", "Std Error", "Rel Error", "MC Error"]
data = zip(num, I_array, standard_error_array, relative_error_array,
↪  montecar_error_array)
print(tabulate(data, headers=headers, tablefmt="grid",
↪  floatfmt="20.18f"))
```

具体输出结果见结论与分析部分。

4.4 数据可视化

要画出相对误差关于 $1/\sqrt{N}$ 变化的折线图，首先要将相应的数据以数组的形式赋值给相应坐标变量，之前以数组的形式输出结果的优势就体现出来了。

```
# 准备绘图数据: x=1/sqrt(N), y= 相对误差
x = 1 / np.sqrt(num)
y = relative_error_array

# 创建画布
plt.figure(figsize=(12, 12), dpi=300)
# 绘制数据点
plt.plot(x, y, "bo-", markersize=8, label="data")
```

然后进行线性度的分析即线性拟合并输出线性回归的结果。

```
# 进行线性回归分析
slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)
r_squared = r_value**2  # 计算 R 平方值

# 生成拟合直线数据
x_fit = np.array([np.min(x), np.max(x)])
y_fit = slope * x_fit + intercept

# 绘制拟合直线
plt.plot(x_fit, y_fit, "r--", linewidth=2, label=f"linear fitting\ny
↪  = {slope:.2f}x + {intercept:.2f}\nR² = {r_squared:.2f}",)

# 添加图例、标签和标题
plt.legend(loc="best")
plt.xlabel("1/sqrt(N)")  # 横坐标: 1/根号 N, 预期与相对误差成线性关系
plt.ylabel("relative error")  # 纵坐标: 相对误差
plt.title("relative error vs. 1/sqrt(N)")  # 标题
plt.grid(True)  # 显示网格

# 显示图形
```

```
plt.show()

# 输出线性回归结果
print(f" 斜率: {slope:.4f}")  # 预期接近理论收敛速率 O(1/sqrt(N)) 的
↪  系数
print(f" 截距: {intercept:.4f}")  # 预期接近 0, 表示系统误差较小
print(f"R 平方: {r_squared:.4f}")  # 接近 1 表示线性关系显著
```

5. 结论与分析

5.1 样本数量为 16 时的积分估计值

样本数量为 16 即蒙特卡洛积分的过程中撒出 16 个点的情况中得到蒙特卡洛积分值为 `26.577936994048187` 。

5.2 样本数量 $N = 1, 2, 4, 8, \cdots, 8192$ 时得到的结果

在样本数量 $N = 1, 2, 4, 8, \cdots, 8192$ 的情况下得到的积分估计值 `Integral` ，标准误差 `Std Error` ，相对误差 `Rel Error` 和蒙特卡洛误差 `MC Error` 如下所示：

N	Integral	Std Error	Rel Error	MC Error
1.000000000000000000	39.809175586907933564	0.000000000000000000	0.541000345299662011	0.000000000000000000
2.000000000000000000	33.414261722208308925	1.323896467234693564	0.293455292472579787	0.936136169570565668
4.000000000000000000	18.450503908144511911	6.088941945619769314	0.285786945491180167	3.044470972809884657
8.000000000000000000	25.220645074895646331	2.936643315201899807	0.023716964842749131	1.038260201052703557
16.000000000000000000	28.693717387686216824	2.475530957083520889	0.110724544039466508	0.618882739270880222
32.000000000000000000	29.311428788102254828	1.555955510012967169	0.134635953087829274	0.275056673088685533
64.000000000000000000	26.805255932151343501	1.244470834097857859	0.037622810276826249	0.155558854262232232
128.000000000000000000	24.406674024128385270	0.773198406955674655	0.055225521646643107	0.068341729595124162
256.000000000000000000	24.906172000561554114	0.574091210298475207	0.035890116107294638	0.035880700643654700
512.000000000000000000	25.627803190672491240	0.407406338342098828	0.007956005522355132	0.018004986533754937
1024.000000000000000000	25.969001276909882137	0.289920354948499936	0.005251662331995484	0.009060011092140623
2048.000000000000000000	25.611925895128123898	0.203787801736402208	0.008570610511169353	0.004503116766590928
4096.000000000000000000	26.046356940227667565	0.146790784340227964	0.008246075105587178	0.002293606005316062

图 2 在样本数量 $N = 1, 2, 4, 8, \cdots, 8192$ 的情况下得到的积分估计值 `Integral` ，标准误差 `Std Error` ，相对误差 `Rel Error` 和蒙特卡洛误差 `MC Error`

可以看到，随着样本数量的增大，各类误差都在大体上呈现逐渐减小的趋势，这与数值积分本身期望的事情相符合。

5.3 相对误差关于 $1/\sqrt{N}$ 变化的折线图与线性拟合

将在样本数量 $N = 1, 2, 4, 8, \cdots, 8192$ 的情况下得到的相对误差 `Rel Error` 与 $1/\sqrt{N}$ 做出相互对应的折线图并进行线性拟合，得到如下所示的结果：

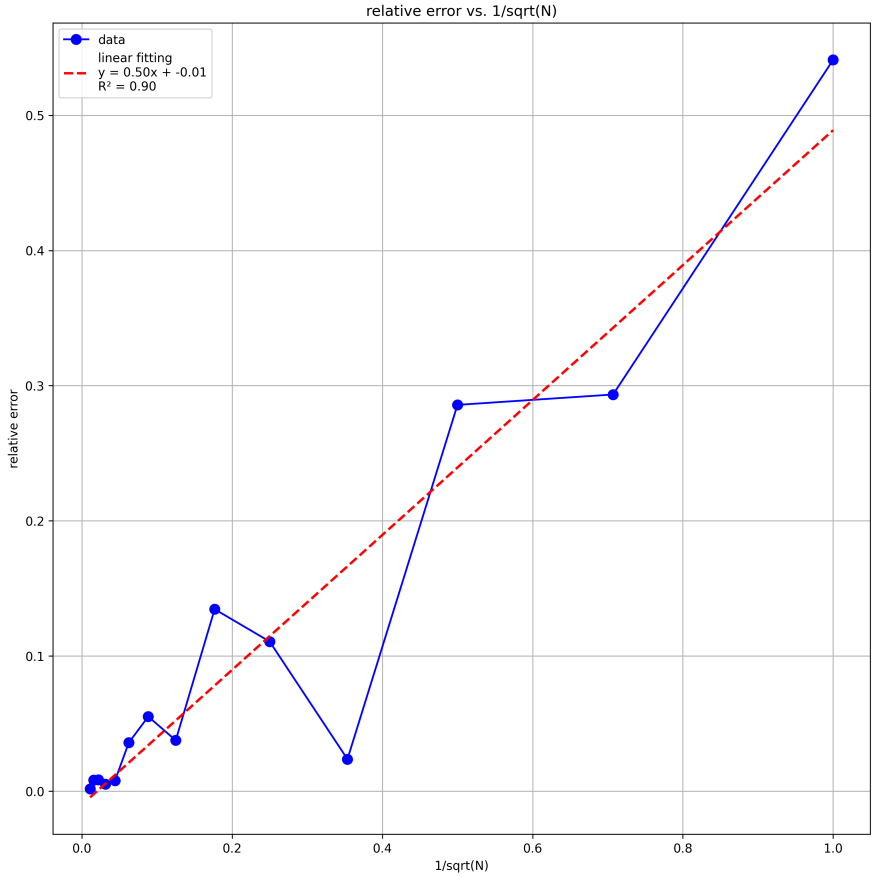

图 3 相对误差关于 $1/\sqrt{N}$ 变化的折线图及线性拟合

相应的线性回归结果如下：

```
斜率: 0.4990
截距: -0.0100
R 平方: 0.9019
```

$R^2 = 0.9019$ 表明相对误差与 $1/\sqrt{N}$ 确有较大的线性关系。蒙特卡洛方法积分接近理论值的收敛速度与 $1/\sqrt{N}$ 是正相关的关系，线性回归结果中给出的截距十分接近 0

恰恰与这一点相符合。

5.4 对于蒙特卡洛积分准确性的分析

根据大数定律和中心极限定理，当样本数量足够大的时候，蒙特卡洛方法对积分的估计值会无限接近于真实值，其相对误差也会相应地无限趋近于 0，即估计的积分值更加准确。从图3就可以看出相对误差是与 $1/\sqrt{N}$ 成线性关系的。回顾蒙特卡洛误差（式4）的定义：

$$\epsilon = V\frac{\sigma_f}{\sqrt{N}} \tag{4}$$

也可以得到同样的结论，因而蒙特卡洛误差是一个很好的蒙特卡洛积分精度的表征标准。

因此，样本数量越多，蒙特卡洛积分的精度就越高，精度可以用蒙特卡洛误差公式来估计。