上机报告

实验名称：

基于 Sympy 库与蒙特卡洛两种方法实现抛物线 $y=x^2$ 与 x 位于[0,1]所围区域面积的求解。

实验目的：

1. 使用 Sympy 库计算抛物线 $y=x^2$ 在区间[0,1]与 x 轴所围区域的精确面积；
2. 使用蒙特卡洛方法估算该区域的面积，并确定获得误差<=0.5 的最小随机样本点数；
3. 展示蒙特卡洛方法中面积值与样本点数变化的趋势。

实验环境：

1. 硬件：

   CPU：Inter Core i9

   GPU：4060

2. 操作系统：

   Windows 11

3. 软件：

   Python 3.8.20

   开发工具：VSCode

   虚拟环境管理：Conda

使用的主要库：Sympy，Numpy，Matplotlib，random

算法描述：

一、使用 Sympy 计算精确面积

1. 定义变量 x 和函数 $y=x^2$；
2. 使用 Sympy 库中积分函数 sp.integrate 对函数在区间[0,1]进行积分；
3. 得到精确面积值

二、使用蒙特卡洛方法估算面积

1. 变量定义：初始化随机采样点的数量 n 与落在范围内的点的数量 k，以及循环计数器 j。
2. 生成随机点(x ,y)，分别处于 0 到 1 区间；
3. 进行循环验证：每次循环对应进行 n 次判断，对于每一对随机生成的（x,y）点，检查其是否满足以下条件：$y<=x^2$,如果满足，计数器 k 增加 1。
4. 每次采样完 n 个点后，使用以下公式估算面积: $S=k/n$.
5. 误差分析与收敛判断：

(1) 平滑计算：每 10 次计算进行一次平均平滑计算，取最近 10 次的面积均值来判断估算值是否收敛到精确值附近。

(2) 误差计算：利用下面公式计算误差：

$$Error=|evaluate\_ans-exact\_ans|$$

其中，evaluate_ans 表示最近 10 次计算面积的均值，exact_ans 表示准确值。

(3) 收敛判断：如果误差小于预设的阈值 0.05，即认为估算值已收敛到精确值，退出循环并输出最终的估算结果;

(4) 样本点数增加与循环：在每次估算之后，如果误差大于阈值，则继续增加样本点数（50 个），直到满足误差条件为止，以保证数据最终的收敛，使结果更合理。

三、**算法核心思路总结：**

蒙特卡洛方法通过随机采样的方式，模拟一个均匀分布的样本点集合，并计算这些点落在函数曲线 y=x^2 下方的比例。根据样本点落在曲线下方的比例，推算出曲线下方的面积。通过不断增加采样点数，逐步提高估算精度，直到误差满足预设条件，从而确保结果的收敛性与合理性。

源代码：

已提交至 CANVAS 平台，名称："上机报告_题目二_赵博文.py"

实验步骤：

一、**准备阶段**

1. 安装必要的 Python 库，主要包括：Sympy，Numpy，Matplotlib，random
2. 配置开发环境：用 Conda 创建虚拟环境，并在 VSCode 上选择对应的 Python 解释器。

二、**使用 Sympy 库计算精确面积**

1. 构建模型：将面积求取问题转化为一定范围内函数的定积分。
2. 定义符号变量与函数：x，y=x^2
3. 计算求解并输出：调用积分函数 integrate 进行计算，得到精确面积。
4. 可视化：调用 Matplotlib 绘制函数 y=x^2 图像，并通过 fill_between 函数标注区域[0,1]下的范围。

三、使用蒙特卡洛方法获得精确面积值（误差<=0.05）的最小随机样本点数

1. 构建模型：将区域面积的求解问题转化为概率问题，即在矩形区域内生成样本点作为总范围，将落在曲线 y=x^2 下方的点的集合作为样本点，计算点落在指定范围的概率作为面积的估算值。
2. 初始化变量：

(1) 设定初始样本点数 n=1，表示开始时每次计算只生成一个样本点。

(2) 定义变量 k 用于计数落入曲线下方的点，j 作为循环计数器。

(3) 初始化变量 ans 用于记录每次估算的面积值，ans_new 用于平滑后的面积值。

```
# 方法2 --> 蒙特卡洛方法
n = 1  # 采样点数初始化
k = 0  # 落入曲线指定范围下的点数初始化
j = 0  # 循环计数器初始化
ans = 0  # 估算面积值
ans_new = 0  # 平均后的估算面积值
y_num = []  # 存储每次计算的估算面积
n_values = []  # 用于记录每次计算时的n值
max_trials = 10000  # 最大循环次数限制
```

3. 进行蒙特卡洛采样

在循环中通过随机生成 n 个点，计算每个点是否落在曲线下方。若落在下方，则增加计数器 k。

```
while j < max_trials:
    k = 0  # 初始化
    for _ in range(n):
        # 随机生成点的坐标
        x_val, y_val = random.uniform(0, 1), random.uniform(0, 1)
        # 判断点是否落在曲线下方,若成立则计数器加1
        if y_val <= x_val**2:
            k += 1
    ans = k / n  # 计算当前估算面积
```

4. 平滑估算结果

为保证随机点数该数据的稳定性和合理性，每增加新的采样点，立刻计算当前的估算面积，并进行平滑（取均值）。

```
# 每 10 次计算平滑一次结果
if len(y_num) >= 10:
    ans_new = np.mean(y_num[-10:])  # 取最近 10 次的均值
```

5. 检查是否收敛

判断估算结果 ans_new（即最近 10 次面积的平均值）是否已经收敛到精确值 ans_s（1/3）附近，若满足误差条件（误差 <= 0.05），则结束循环并记录当前的样本数作为最小样本点数。

6. 增加采样点数

如果未达到误差要求，则增加采样点数 n（为确保数据的合理性，在此每次增加 50 次），并进行下一轮计算。

```
# 判断估算值是否收敛到精确值附近
if abs(ans_new - ans_s) <= 0.05:
    final = ans_new  # 保存最终的估算值
    final_num = n   # 保存最终使用的采样点数
    break
else:
    n += 50  # 每次循环增加采样点数
    j += 1  # 循环计数器加 1
```

7. 最终输出

输出最小采样点数（final_num=n）与最终估算面积。

```
print(f"最少次数为 {final_num}，估算的面积为 {final:.5f}")
```

8. 可视化

绘制蒙特卡洛估算值与样本点数的关系图，横坐标为每次参与计算的 n 值，纵坐标为当前 n 值估算的面积，并用红色虚线标注精确值以方便观察收敛趋势。

```
plt.figure(figsize=(10, 6))
plt.plot(n_values, y_num)
plt.axhline(float(ans_s), color="red", linestyle="--",
            label="Exact Value (Sympy)")
plt.title("Monte Carlo Method: Evaluate Area")
plt.xlabel("n")
plt.ylabel("S_Evaluate")
plt.show()
```

实验结果

一、精确面积计算

基本原理：$$\text{精确面积} = \int_0^1 x^2\,dx = \frac{1}{3} \approx 0.3333$$

```
x**2函数与x从0到1所围区间的精确面积是1/3
```

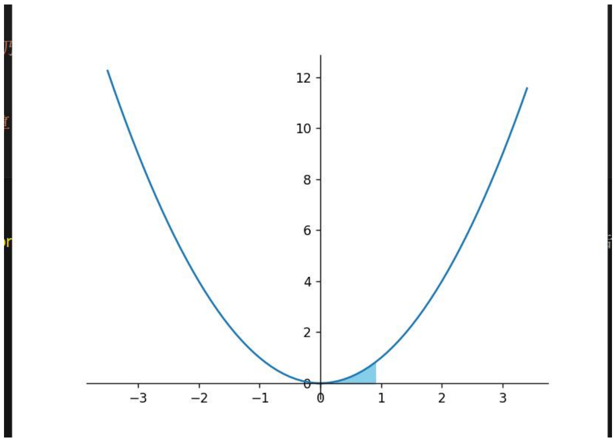
结果表明，计算结果与实际一致。

二、画图展示所围区域

利用 Matplotlib 绘制函数 y=x^2 的曲线，并标注曲线下方的目标区域（[0,1] 区间）。

图中，浅蓝色阴影区域直观展示了所需计算的面积，强调了抛物线与

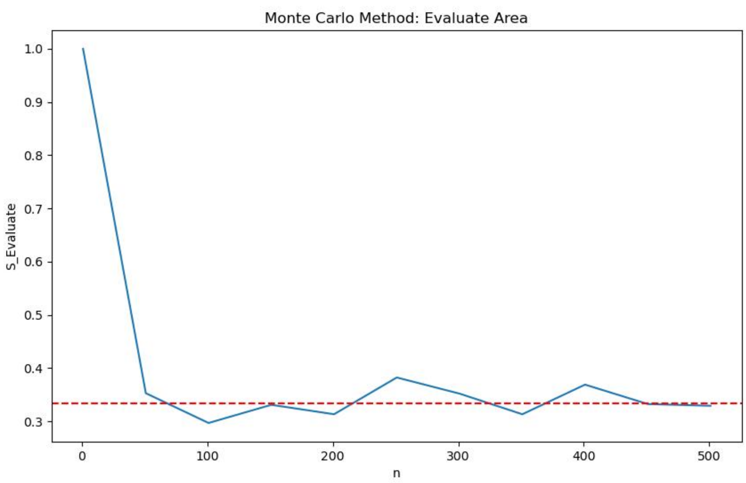
x 轴所围成的面积范围。



三、计算蒙特卡洛方法获得的最小随机样本点数与估算结果

最少样本点为：501 ，估算面积为：0.33736

四、展示蒙特卡洛方法获取的面积值随机样本点变化的趋势图

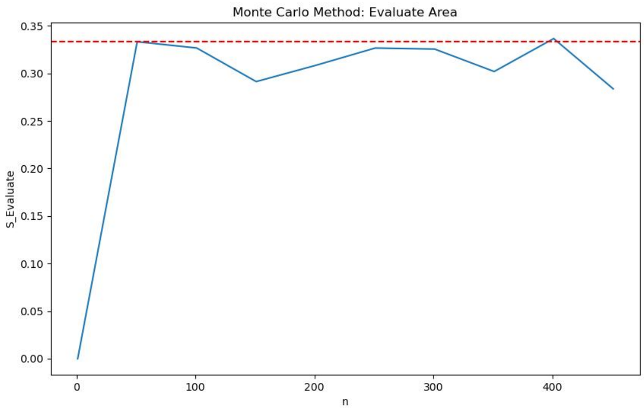


趋势图清晰展示了以下特性：

1. 随着样本点数 n 的增加，蒙特卡洛估算面积逐渐逼近精确值 1/3 且波动幅度显著减小，表明数据逐渐收敛。
2. 当样本点数不足时，估算面积存在较大波动，未能满足误差要求。

3. 当样本点数达到 501 时，数据波动基本消失，估算结果稳定且收敛性良好。

由于样本点的随机性，每次运行得到的答案与图像并不唯一，主要出现两种答案：501 个与 451 个，而 451 个样本点对应的图像为：



可见，451 个样本点的数据收敛性不如 501 个，数据欠缺严谨性，因此最后采用了最少样本点为 501 个。

对于 501 个样本点数据，最后的估算面积值为 0.33736，十分接近准确值，并且图像数据最后通过平滑计算（每 10 次采样平滑一次），提高估算结果的稳定性，收敛性越来越好，因此该数据具有合理性与严谨性。

问题讨论

一、实验过程中遇到的问题与解决方案

在进行蒙特卡洛方法计算时，一开始我的代码如下：

```python
while 1:
    k = 0
    for _ in range(n):
        x_val, y_val = random.uniform(0, 1), random.uniform(0, 1)
        if y_val <= x_val**2:
            k += 1
    ans = k / n
    if abs(ans-ans_s) <= 0.05:
        print(f"最少样本点数为{n}")
        break
    else:
        n += 1
```

而在实际运行过程中，我发现最终结果往往不超过 10 个，这并不符合蒙特卡洛方法的规律，经过思考我发现这个算法有一处缺陷，即只要得到一次符合误差范围的结果便跳出循环。而蒙特卡洛方法所产生的样本点是随机的，按照此方法得到的结果只是一次小概率事件，数据并没有收敛，也就是数据最后并未趋近于精确值 1/3。

因此，为了确保数据的收敛，我在判断之前对计算结果进行了达到 10 次面积计算后，对最后 10 次的面积进行平均取值，以该平均值为依据判断是否满足误差要求，同时每次取样的数量 n 递增 50。这样，通过平滑计算，避免了数据的偶然性。

为什么选取最近 10 次的面积进行平滑呢？对于稳定性而言，蒙特卡洛方法的随机性会导致估算结果波动，10 次的面积平滑有助于平衡波动，使计算结果更加稳定；对于时间与效率而言，能够在一定程度上减少计算量的同时，确保估算值的准确性。如果选择过小的数目（例如每 2 次计算一次），可能会增加计算次数，且波动较大；如果选择过大的数目（例如每 100 次计算一次），可能会延迟估算结果的收敛。

为什么选取 50 作为样本点 n 每次的递增数量？对于科学性而言，蒙特卡洛方法的精度与采样点数成正比，随着采样点数 n 的增加，

估算的精度会逐渐提高，因为更多的采样意味着更多的样本数据可以用于推测真实的面积值。递增 50 可以确保每次循环都在一定程度上增加精度，同时也不会增加过多的计算负担；对于计算效率与精度而言，增加采样点数有助于提高估算精度，但如果每次增加太多（例如递增 100 或 200），会导致计算负担过重，尤其是在最大迭代次数 max_trials 设定为 10000 时。递增 50 是一个合理的折中，它既能有效提高估算精度，又能控制计算时间。

对于两个关键参数的调整，针对不同问题求解也许会有更好的选择，而目前要进行的面积估算问题，经过多次探索验证，这两个数据在科学性和效率两方面兼具优势，因此采用 10 与 50。

实验心得

在实验过程中，我经历了分析问题、抽象模型、调用方法、优化算法、验证调试、反思总结这几个阶段，对于实验有了更加清晰的认知。

在得出不合理数据之后，我通过反思蒙特卡洛算法的核心要点，抓住了其样本点随机性的特征以及数据应收敛的必要条件，重新审视算法，进行了更改和验证。通过反复的调试，最终得到合理的结果。

对于本次的实验，我对自己的实验过程较为满意，得到的实验效果也很好。此次实验不仅是对蒙特卡洛算法的加深理解和编程语言的熟悉，更重要的是对数据的分析和反思，以及优化计算过程的方法，不断反思验证，才能得到合适的实验数据。