计算方法（A）

实验（三） 实现龙贝格积分

姓　　名	熊恪峥
学　　号	22920202204622
日　　期	2022年4月6日
学　　院	信息学院
课程名称	计算方法（A）

实验（三） 实现龙贝格积分

目录

1 实现

1.1 龙贝格积分法

龙贝格积分法是通过外推法不断使用梯形公式来提高计算精度的数值积分法，它在区间上取等距离的点。它的过程如下：

1.2 文件结构

本程序实现了龙贝格积分以及用于基准测试和比较的蒙特卡洛积分法和拟蒙特卡洛（Quasi-Monte Carlo）积分法。主要文件和作用如表格 1.2所示。

文件	作用
integral/romberg.py	龙贝格积分实现
integral/qmc.py	拟蒙特卡洛积分法实现
integral/montecarlo.py	蒙特卡洛积分法实现

表格 1: 主要文件和作用

本程序依赖如下两个第三方库：

- **Numpy**: 用于加速向量运算
- **Scipy**: 用于生成低差异序列（Low-discrepancy sequence）

1.3 接口实现

本程序为龙贝格积分法实现以下接口，它支持通过给定任意可调用对象（Callable）表示的函数进行积分，具有良好的可扩展性。

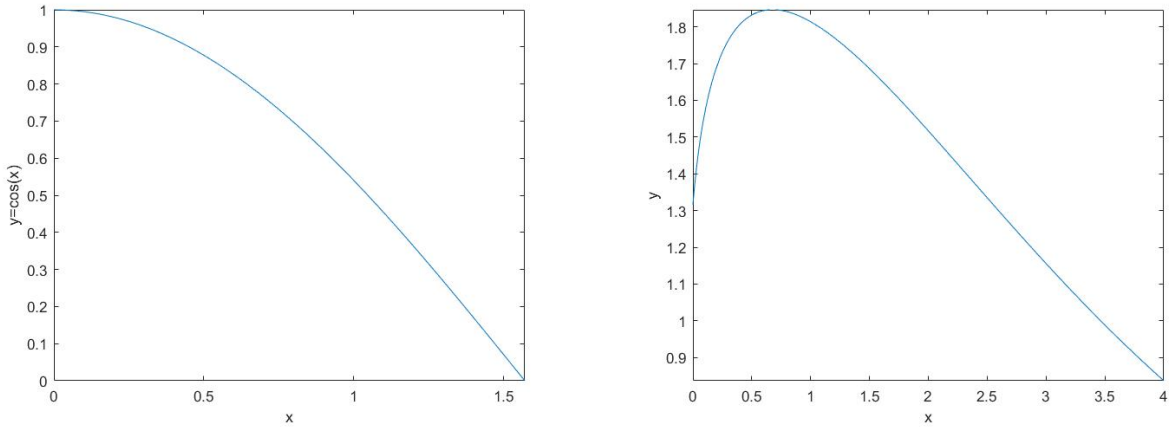
```
1 def integrate(f: Callable[[np. float], np. float], a: np. float, b: np. float, epsilon: np.
      float = 0.001,
2               n:  int = 32) -> np. float:
```

该接口有如下参数：

- **f**: 被积函数的可调用对象。
- **a**, **b**: 表示积分区间$[a, b]$。
- **epsilon**: 当两次迭代数值变化$\Delta T \leq \varepsilon$时停止迭代，可以控制积分精度。
- **n**: 最大迭代次数。

2 基准测试

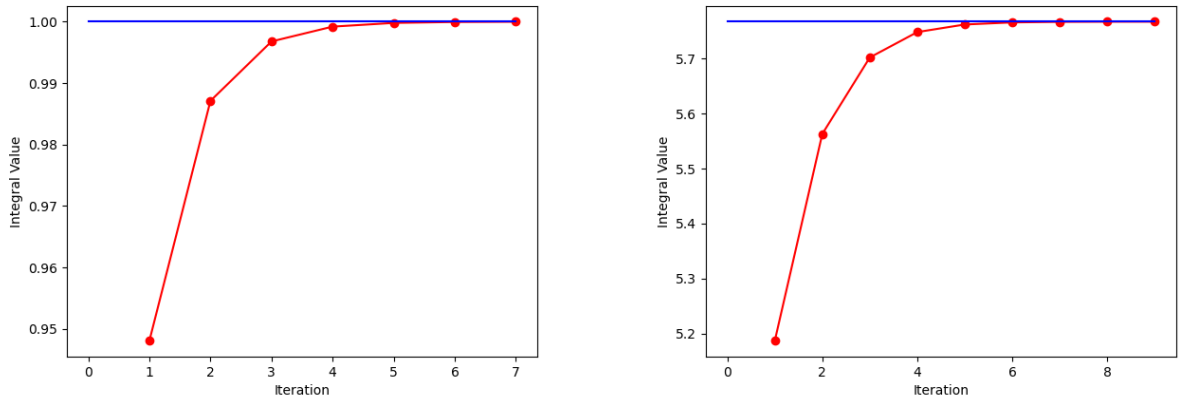
为了测试龙贝格积分法的效果，以Matlab 2021b的积分值为真值、蒙特卡洛积分法和拟蒙特卡洛积分法为基线进行测试，计算积分$\int_0^{\frac{\pi}{2}} \cos x \, \mathrm{d}x$和$\int_0^4 \sqrt[4]{15x^3 + 21x^2 + 41x + 3} \cdot e^{-0.5x} \, \mathrm{d}x$。被积函数的图形如图 1



(a) 函数$f(x)=\cos x$ (b) 函数$g(x)=(15x^3+21x^2+41x+3)^{\frac{1}{4}}\cdot e^{-0.5x}$

图 1: 被积函数图象

调用本程序实现的龙贝格积分法函数，要求精度0.0001，迭代次数最多128次，可以得到每次迭代的积分值如图 2。可以发现龙贝格积分法的收敛速度很快，约3至4次迭代即可得到和真实值相近的积分值。



(a) 积分$\int_0^{\frac{\pi}{2}} f(x)\,\mathrm{d}x$迭代过程 (b) 积分$\int_0^4 g(x)\,\mathrm{d}x$迭代过程

图 2: 被积函数图象

龙贝格积分法积分得出的数据如表 2所示。在迭代次数10以内的条件下积分值的精确度已经达到了极高的水平，相对误差对于两个不同的函数均小于0.001%。这说明了龙贝格法有良好的精度。

函数	测量值	参考值（以Matlab 2021b为基准）	相对误差(%)
$f(x)$	0.9999874501175262	1	0.0012
$g(x)$	5.767412519408859	5.767433490695931	0.00036

表格 2: 龙贝格积分法积分值

为了测量该实现的运行效率，

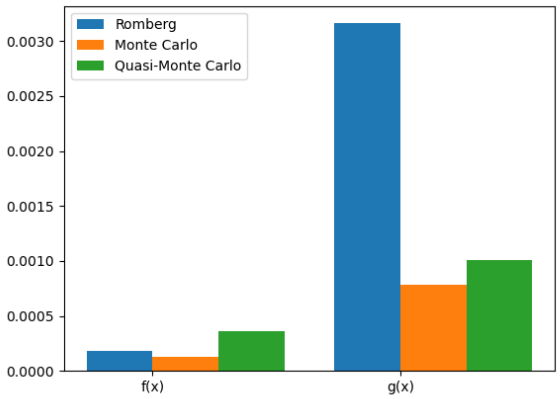
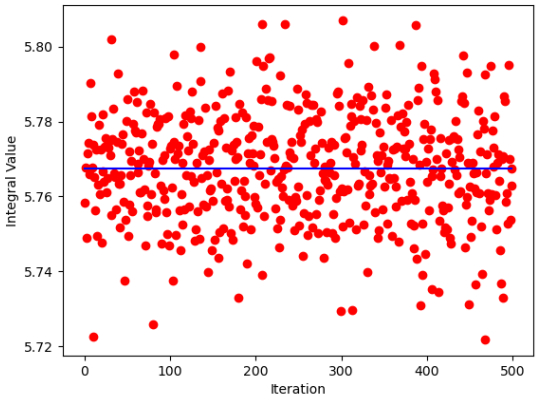
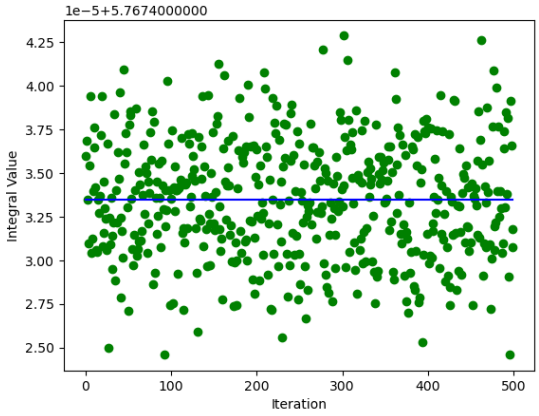


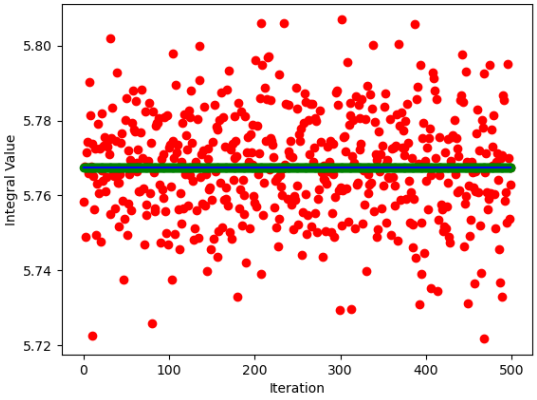
图 3: 平均运行时间



(a) 蒙特卡洛法500次分布



(b) 拟蒙特卡洛法500次分布



(c) 同坐标系显示

图 4: 蒙特卡洛和拟蒙特卡罗的分布情况

3 结论

导致龙贝格积分法的实现效率相对一般的原因可能是因为这个递推过程很难被向量化优化，这导致了该

算法	均值	方差
蒙特卡洛法	5.766607543456825	0.00018877067621830997
拟蒙特卡洛法	5.767433680428228	$1.0746874157286956 \times 10^{-11}$

表格 3: 随机方法的均值与方差

实现没有有效利用现代硬件的优化。但是这造成的性能差异并不显著，可见龙贝格积分法是一种非常高效的算法。

尽管在为了达到与蒙特卡洛积分法特别是使用了拉丁超立方采样的拟蒙特卡洛积分法可以比拟的精度，需要较多的迭代次数从而影响了运行效率，但是考虑到拉丁超立方采样在多变量积分和图形渲染领域的大量应用侧面证明的高效性和准确性，这也是正常的。

A 附录：蒙特卡洛积分法和拟蒙特卡洛积分

使用蒙特卡洛积分是一种采用蒙特卡洛法估计积分的方法。设有积分(1)

$$I = \int_a^b f(x) d\,\mathrm{d}x \tag{1}$$

那么可以使用(2)估计这个积分。其中$pdf(X)$是概率密度函数。特别地在实现中常取均匀分布$X \sim \mathcal{U}(a,b)$，则$pdf(x) = 1/(b-a)$。

$$\bar{I_n} = F_n(X) = \frac{1}{n}\sum_{k=1}^{n}\frac{f(X_k)}{pdf(X_k)} \tag{2}$$

Proof. 要证明这个估计是正确的，就需要证明该估计的数学期望和积分值相等。

$$\begin{aligned} E[F_n] &= E\left[\frac{1}{n}\sum_{k=1}^{n}\frac{f(X_k)}{pdf(X_k)}\right] \\ &= \frac{1}{n}\sum_{k=1}^{n}\int \frac{f(x)}{pdf(x)}\cdot pdf(x)\,\mathrm{d}x \\ &= \frac{1}{n}\sum_{k=1}^{n}\int f(x)\,\mathrm{d}x \\ &= \int f(x)\,\mathrm{d}x \end{aligned}$$

□

根据以上结论，可以得到算法 1

算法 1 蒙特卡洛法积分

Input: 函数F，区间$[a,b]$，点数n

```
procedure INTEGRATE(F, a, b, n)
    X = [x_1, x_2, x_3, ...], x_i ~ U(a, b)
    Y = F(X)
    return (b − a) (1/n) Σ_{i=1}^{n} Y_i
```

作为一种随机算法，实际上在n较大时有较好的精度，它的收敛速度是$\mathcal{O}(n^{-0.5})$ 如果采用低差异序列替换纯随机数，积分收敛速度能够达到$\mathcal{O}(n^{-1})$，而且能够避免随机采样中采样点的聚集。一种简单的一维低差异序列是*Van Der Corput*序列。设有b进制数 $x = \sum_{k=0}^{m} d_k \cdot b^k$，则对应的Van der Corput序列可由(3)计算。

$$g_b(x) = \sum_{k=0}^{m} d_k \cdot b^{-k-1} \tag{3}$$

$g_b(x)$满足$g_b(x) \in [0,1]$，因此通过区间映射并替代算法 1中随机数可以得到拟蒙特卡洛积分算法。此外，还可以使用Latin Hypercube Sampling（拉丁超立方采样）来替代随机数。

B 附录：测试代码