习题19.1

用蒙特卡罗积分法求

$$\int_{-\infty}^{\infty} x^2 \exp\left(-\frac{x^2}{2}\right) \mathrm{d}x \tag{1}$$

取概率密度函数为标准正态分布的密度函数 $p(x)=\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)$，则

$$f(x)=\frac{h(x)}{p(x)}=\frac{x^2\exp\left(-\frac{x^2}{2}\right)}{\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)}=\sqrt{2\pi}x^2 \tag{2}$$

将原函数积分表示为函数 $f(x)$ 关于概率密度函数 $p(x)$ 的数学期望

$$\int_{\mathcal{X}} x^2 \exp\left(-\frac{x^2}{2}\right)\mathrm{d}x=\int_{\mathcal{X}} f(x)p(x)\mathrm{d}x=E_{p(x)}[f(x)] \tag{3}$$

- 用编程实现正态分布的抽样函数

```python
import numpy as np
class MonteCarloIntegrator:
    def __init__(self, target_function, sampling_distribution):
        """
        初始化蒙特卡罗积分类
        @params:
        - target_function: function, 所求期望的函数 f(x)
        - sampling_distribution: function, 抽样分布的概率密度函数 p(x)
        """
        self.target_function = target_function
        self.sampling_distribution = sampling_distribution
    def integrate(self, num_samples):
        """
        执行蒙特卡罗积分
        @params:
        - num_samples: int, 抽样样本数量
        @return:
        - float, 样本的函数均值（积分的估计值）
        """
        samples = self.sampling_distribution(num_samples)
        # 对目标函数进行矢量化处理以便对样本集进行操作
        vectorized_function = np.vectorize(self.target_function)
        function_values = vectorized_function(samples)
        # 计算样本的函数值的均值
        return np.sum(function_values) / num_samples

def target_function(x):
    """
    目标函数
```

```
    @params:
    - x: float, 输入值
    @return:
    - float, 函数值 f(x) = x^2 * sqrt(2 * pi)
    """
    return x ** 2 * np.sqrt(2 * np.pi)

def sampling_distribution(num_samples):
    """
    抽样分布的概率密度函数
    @params:
    - num_samples: int, 样本数量
    @return:
    - np.ndarray, 抽样样本数组
    """
    return np.random.standard_normal(int(num_samples))

# 创建蒙特卡罗积分器实例
integrator = MonteCarloIntegrator(target_function, sampling_distribution)

# 执行积分
result = integrator.integrate(10000000)

print("积分结果:", result)
```

```
积分结果: 2.508571638914768
```

习题19.4

验证具有以下转移概率矩阵的马尔可夫链是不可约的，但是周期性的。

$$P = \begin{bmatrix} 0 & 1/2 & 0 & 0 \\ 1 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 1 \\ 0 & 0 & 1/2 & 0 \end{bmatrix} \tag{4}$$

设平稳分布为 $\pi = (x_1, x_2, x_3, x_4)^{\mathrm{T}}$，则由引理19.1可得：

$$\begin{aligned} & x_1 = \frac{1}{2} x_2 \\ & x_2 = 1 x_1 + \frac{1}{2} x_3 \\ & x_3 = \frac{1}{2} x_2 + 1 x_4 \\ & x_4 = \frac{1}{2} x_3 \\ & x_1 + x_2 + x_3 + x_4 = 1 \\ & x_i \geqslant 0, \quad i = 1,2,3,4 \end{aligned} \tag{5}$$

解方程组，得到唯一的平稳分布

$$\pi = (\frac{1}{6}, \frac{1}{3}, \frac{1}{3}, \frac{1}{6})^T \tag{6}$$

- 平稳分布各项均大于0，马尔可夫链从任意状态出发，经过若干步之后，可以到达任意状态，故该马尔可夫链是不可约的。
- 马尔可夫链的转移仅发生在相邻奇偶状态之间，从每个状态出发，返回该状态的时刻都是2的倍数：$2, 4, 6, \cdots, 2n, n \in N^*$。故该马尔可夫链是周期性的，其周期为2.

习题19.7

假设进行伯努利实验，后验概率为 $P(\theta|y)$，其中变量 $y \in \{0, 1\}$ 表示实验可能的结果，变量 θ 表示结果为 1 的概率。再假设先验概率 $P(\theta)$ 遵循 Beta 分布 $B(\alpha, \beta)$，其中 $\alpha = 1, \beta = 1$；似然函数 $P(y|\theta)$ 遵循二项分布$\text{Bin}(n, k, \theta)$，其中 $n = 10, k = 4$，即实验进行 10 次其中结果为 1 的次数为 4。试用 Metropolis-Hastings 算法求后验概率分布 $P(\theta|y) \propto P(\theta)P(y|\theta)$ 的均值和方差。（提示：可采用 Metropolis 选择，即假设建议分布是对称的。）

后验概率 $P(\theta) \propto B(1, 1)$，后验函数 $P(y|\theta) \propto \text{Bin}(10, 4, \theta)$

则后验概率分布

$$\begin{aligned} P(\theta|y) &\propto P(\theta)P(y|\theta) \\ &\propto B(1,1)\text{Bin}(10, 4, \theta) \end{aligned} \tag{7}$$

- 取接受分布为 $\text{Bin}(10, 4, \theta)$，取建议分布为 $B(1, 1)$

第3步：自编程实现 Metropolis-Hastings 算法求解

```python
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import beta, binom

class MetropolisHastings:
    def __init__(self, proposal_dist, target_dist, burn_in=1000, num_samples=10000):
        """
        Metropolis Hastings

        :param proposal_dist: 建议分布
        :param target_dist: 目标分布
        :param burn_in: 收敛步数
        :param num_samples: 样本数量
        """
        self.proposal_dist = proposal_dist
        self.target_dist = target_dist
        self.burn_in = burn_in
        self.num_samples = num_samples

    @staticmethod
    def __calc_acceptance_ratio(q, p, x, x_prime):
        """
        计算接受概率

        :param q: 建议分布
```

```
        :param p: 目标分布
        :param x: 当前状态
        :param x_prime: 候选状态
        :return: 接受概率
        """
        prob_1 = p.prob(x_prime) * q.joint_prob(x_prime, x)
        prob_2 = p.prob(x) * q.joint_prob(x, x_prime)
        alpha = min(1., prob_1 / prob_2)
        return alpha

    def sample(self):
        """
        Metropolis Hastings 算法采样
        :return: 样本数组、样本均值、样本方差
        """
        all_samples = np.zeros(self.num_samples)
        x_0 = np.random.random()

        for i in range(self.num_samples):
            x = x_0 if i == 0 else all_samples[i - 1]
            x_prime = self.proposal_dist.sample()
            alpha = self.__calc_acceptance_ratio(self.proposal_dist, self.target_dist, x, x_prime)
            u = np.random.uniform(0, 1)

            if u <= alpha:
                all_samples[i] = x_prime
            else:
                all_samples[i] = x

        samples = all_samples[self.burn_in:]
        dist_mean = samples.mean()
        dist_var = samples.var()
        return samples, dist_mean, dist_var

    @staticmethod
    def visualize(samples, bins=50):
        """
        可视化展示
        :param samples: 抽取的样本集合
        :param bins: 直方图的分组个数
        """
        plt.figure(figsize=(10, 6))
        plt.hist(samples, bins, density=True, alpha=0.7, color='blue', edgecolor='black')
        plt.title('Metropolis Hastings Sample Distribution')
        plt.xlabel('Value')
        plt.ylabel('Frequency')
        plt.xlim(0, 1)
        plt.grid(True)
        plt.show()
```

```
class ProposalDistribution:
    """
    建议分布
    """

    @staticmethod
    def sample():
        """
        从建议分布中抽取一个样本
        :return: 样本值
        """
        return beta.rvs(1, 1)

    @staticmethod
    def prob(x):
        """
        P(X = x) 的概率
        :param x: 样本值
        :return: 概率
        """
        return beta.pdf(x, 1, 1)

    def joint_prob(self, x_1, x_2):
        """
        P(X = x_1, Y = x_2) 的联合概率
        :param x_1: 样本值1
        :param x_2: 样本值2
        :return: 联合概率
        """
        return self.prob(x_1) * self.prob(x_2)
class TargetDistribution:
    """
    目标分布
    """

    @staticmethod
    def prob(x):
        """
        P(X = x) 的概率
        :param x: 样本值
        :return: 概率
        """
        return binom.pmf(4, 10, x)
```

```
import warnings
warnings.filterwarnings("ignore")

# 参数设置
```

```python
burn_in = 1000
num_samples = 10000

# 创建建议分布和目标分布
proposal_dist = ProposalDistribution()
target_dist = TargetDistribution()

# Metropolis-Hastings 算法实例
mh = MetropolisHastings(proposal_dist, target_dist, burn_in, num_samples)

# 采样
samples, dist_mean, dist_var = mh.sample()
print("均值:", dist_mean)
print("方差:", dist_var)

# 可视化结果
mh.visualize(samples, bins=50)
```

```
均值: 0.4193854865801598
方差: 0.018715389639405326
```