

**《随机信号分析课内实验》实验报告册**

**得分：**

# 班级/学号

**姓 名**

# 基础实验一 随机信号的仿真与分析

一、实验内容简介

**1、实验目的：**学习利用计算机进行随机信号仿真的基本方法

**2、实验环境：**

1. 便携式计算机：宏碁传奇X
   1. CPU： Razen 7 5800U
   2. GPU： Nvidia GeForce RTX 3050
2. 系统环境：Windows10 系统家庭版
3. 软件环境：Matlab 2021a

**3、实验原理、实验内容及要求：**

1. 实验原理：对随机信号进行抽样，当抽样的数据足够多，采样的时间足够长时，由于基于随机过程的各态历经性，我们就可以通过计算这一抽样数据的数学特征来实现对这个随机信号的数字特征的计算。
2. 实验内容及要求：

使用matlab语言编程并仿真以下内容

* 1. 产生的10000个泊松分布随机数，计算它们的均值、方差和概率密度、功率密度，自相关函数，并绘出函数曲线。
  2. 产生100个服从的随机数，并计算它们的概率密度，功率谱密度，自相关函数，并绘出函数曲线。
  3. 产生的指数分布随机数，计算它们的概率密度，功率谱密度，自相关函数，并绘出函数曲线。
  4. 产生均匀分布的随机数，计算它们的概率，功率谱密度，自相关函数，并绘出函数曲线。

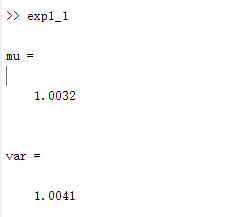
二、实验内容实现

1. 产生对应对随机数：在matlab中有许多不同分布的随机数生成函数。
   1. poissrnd(lambda,sz1,...,szN) 生成泊松分布的随机数，lambda是泊松分布的参数，其中sz1,...,szN表示每个维度的大小。
   2. normrnd(mu,sigma,sz1,...,szN)) 从均值参数为 mu 和标准差参数为 sigma 的正态分布中生成随机数组，其中 sz1,...,szN 指示每个维度的大小。
   3. exprnd(mu,sz1,...,szN) mu是指数分布的参数，根据指数分布生成一个随机数数组，其中sz1,...,szN表示每个维度的大小。
   4. rand(sz1,...,szN) 返回由在区间 (0,1) 内均匀分布的随机数组成的 sz1×...×szN 数组，其中 sz1,...,szN 指示每个维度的大小。
2. 计算均值和方差
   1. 均值：mean(A) 返回A中元素均值。
   2. 方差：var(A) 计算A的方差，如果A是一个向量，则方差是一个标量。
3. 概率密度
   1. 对于离散随机变量，可以通过histcounts(X)进行统计，得到不同数据的数值，在除以总的样本数，就能得到对应的概率密度。
   2. 对于连续随机变量，可以通过ksdensity(x)进行核心平滑密度估计，就能得到对应的概率密度。
4. 功率谱密度：可以使用一下三种方法进行求解
   1. Periodogram函数：通过周期图法 求解功率谱密度。周期图法是指：为得到功率谱估值，先取信号序列的离散傅里叶变换，然后取其幅频特性的平方并除以序列长度N。
   2. pwelch函数：通过Welch方法求解。Welch方法是一种修正周期图功率谱密度估计方法，它通过选取的窗口对数据进行加窗处理，先分段求功率谱之后再进行平均。
   3. 利用维纳—辛钦定理，对自相关函数进行傅里叶变换，即可得到功率谱密度
5. 自相关函数：使用xcorr函数即可进行求解。但是需要注意的是，直接调用xcorr函数进行计算会出现一些问题，在第四部分（遇到的问题及解决的方法）中我会进行详细描述。

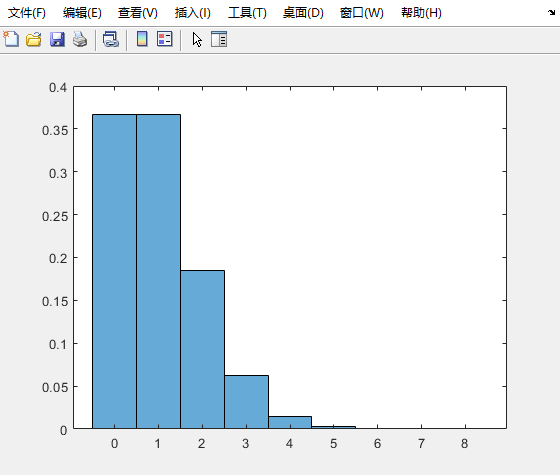
三、实验仿真结论及分析

下面是各个随机数的各种仿真实验结果

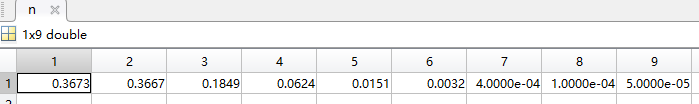
1. 泊松分布
   1. 均值与方差：可以看到与理论值相差甚小。



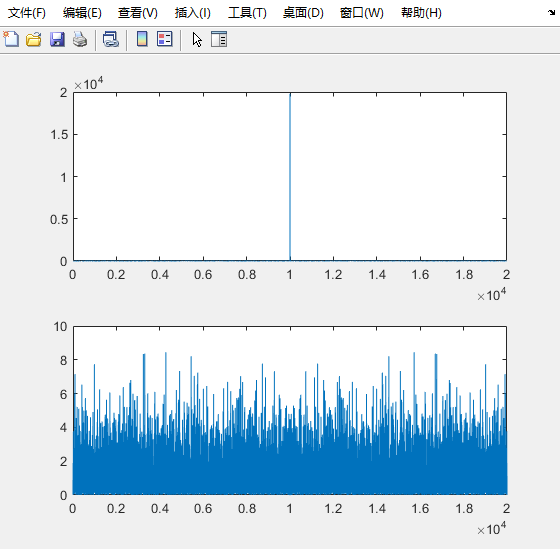
* 1. 概率密度：通过histogram函数，并使用'Normalization','pdf'参数，可以将原本为数值的直方图归一化到概率的格式。得到如下的概率密度图。



如果希望得到数组形式的概率密度，则使用histcount函数，并将得到的频次除以总数，就能得到对应的频率。

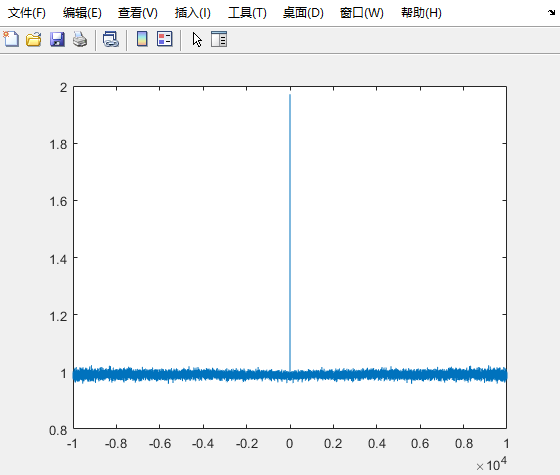


* 1. 功率谱密度：利用维纳-辛钦定理，对自相关函数进行傅里叶变换，得到如下结果。



上图由于信号含有直流分类，导致在零频率处有一个冲激，使得其他部分在图上无法显示出来，所以在下图我将零频附近的冲击消去，重新画图，可以看出在各个频率上基本均匀分布没有明显的分布特征，属于白噪声。

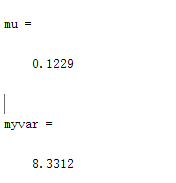
* 1. 自相关函数：使用xcorr函数进行计算。需要注意在第四部分中提到的问题



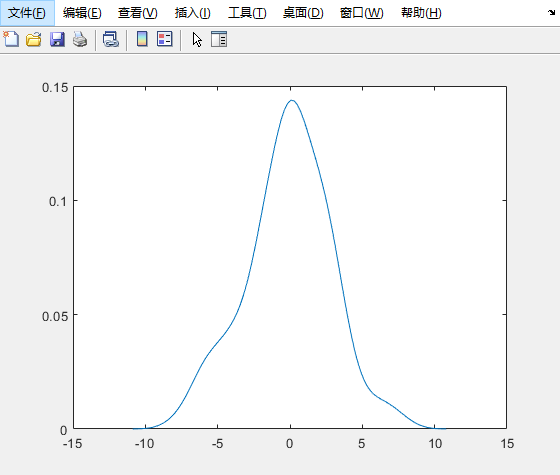
1. 正态分布

由于题目中要求只能生成100个随机数，所以各项数据可能波动较大。

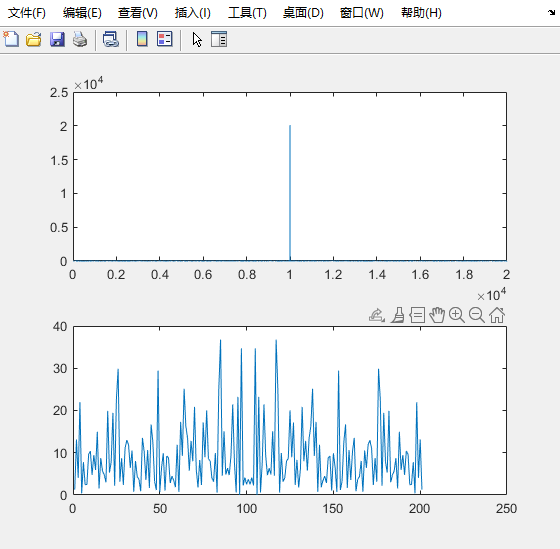
* 1. 均值与方差：可以看到由于数据量小，与理论值还有较大的差异。



* 1. 概率密度：通过ksdensity函数，可通过核函数平滑估计法得到概率密度如下图所示。



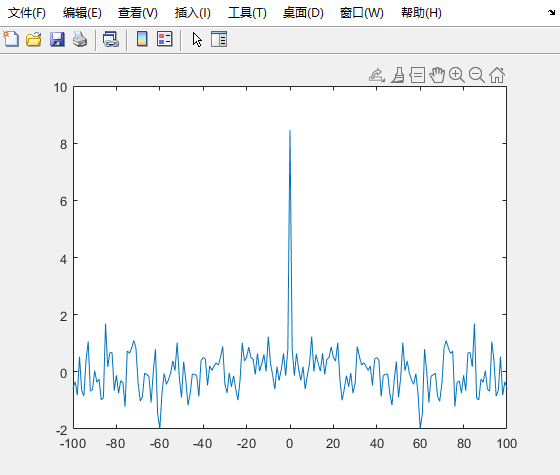
* 1. 功率谱密度：利用维纳-辛钦定理，对自相关函数进行傅里叶变换，得到如下结果。



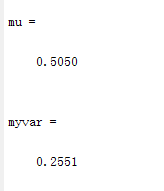
上图由于信号含有直流分类，导致在零频率处有一个冲激，使得其他部分在图上无法显示出来，所以在下图我将零频附近的冲击消去，重新画图，可以看出在各个频率上基本均匀分布没有明显的分布特征，属于白噪声。

同时由于数据量较小，看起来较为稀疏

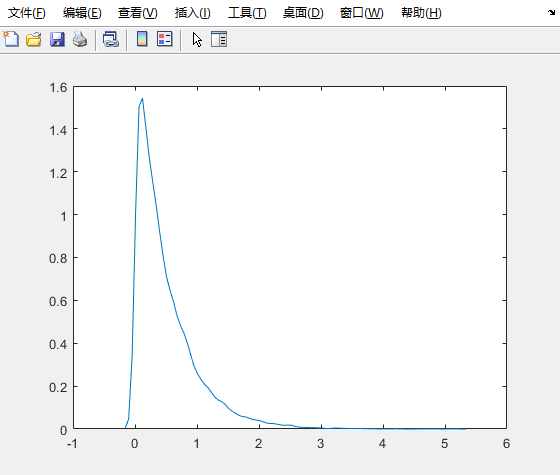
* 1. 自相关函数：使用xcorr函数进行计算。需要注意在第四部分中提到的问题



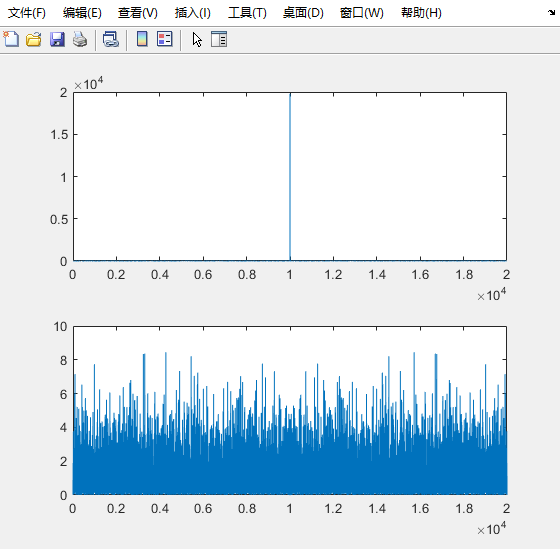
1. 指数分布
   1. 均值与方差：可以看到与理论值相差甚小。



* 1. 概率密度：通过ksdensity函数，可通过核函数平滑估计法得到概率密度如下图所示。

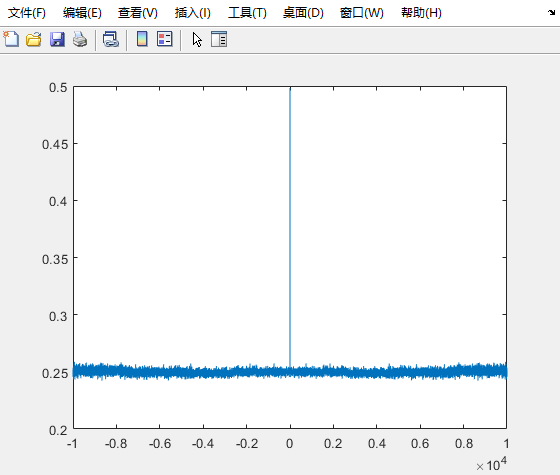


* 1. 功率谱密度：利用维纳-辛钦定理，对自相关函数进行傅里叶变换，得到如下结果。

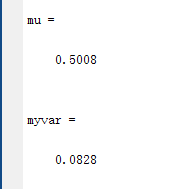


上图由于信号含有直流分类，导致在零频率处有一个冲激，使得其他部分在图上无法显示出来，所以在下图我将零频附近的冲击消去，重新画图，可以看出在各个频率上基本均匀分布没有明显的分布特征，属于白噪声。

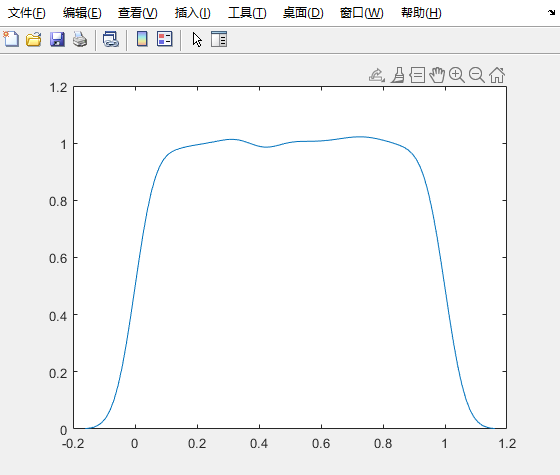
* 1. 自相关函数：使用xcorr函数进行计算。需要注意在第四部分中提到的问题



1. 均匀分布
   1. 均值与方差：可以看到与理论值相差甚小。

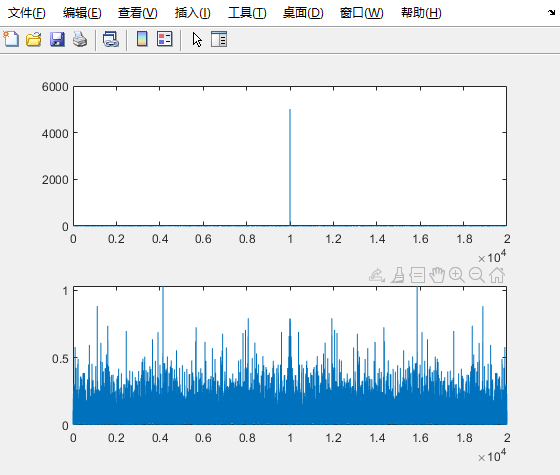


* 1. 概率密度：通过ksdensity函数，可通过核函数平滑估计法得到概率密度如下图所示。



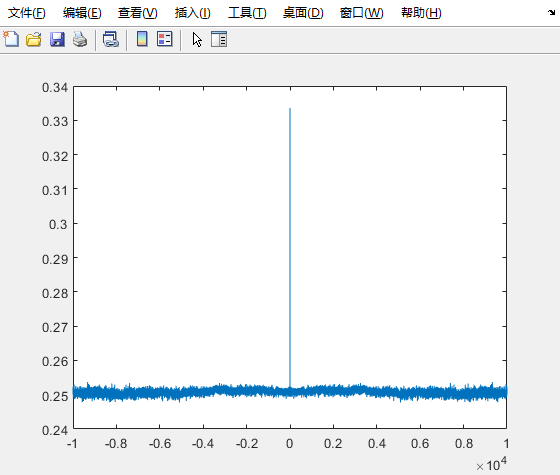
由于核函数平滑估计法的局限性，在两端有些失真。

* 1. 功率谱密度：利用维纳-辛钦定理，对自相关函数进行傅里叶变换，得到如下结果。



上图由于信号含有直流分类，导致在零频率处有一个冲激，使得其他部分在图上无法显示出来，所以在下图我将零频附近的冲击消去，重新画图，可以看出在各个频率上基本均匀分布没有明显的分布特征，属于白噪声。

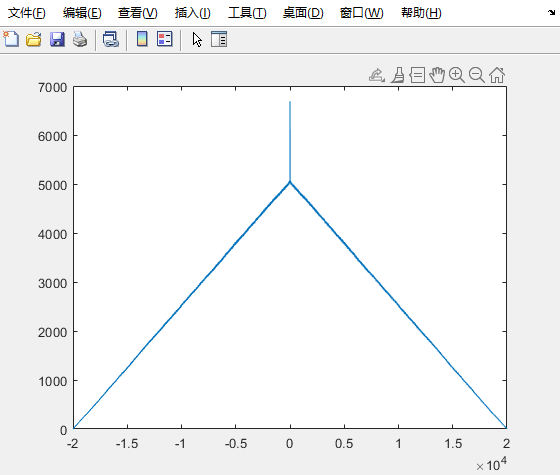
* 1. 自相关函数：使用xcorr函数进行计算。需要注意在第四部分中提到的问题



四、遇到的问题及解决的方法

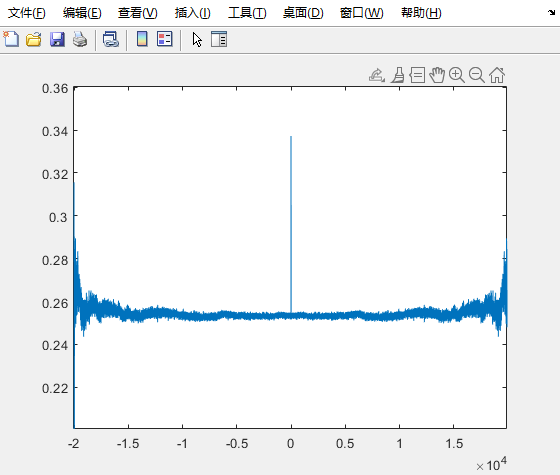
1. 在求解自相关函数时

由于信号本身的长度是有限的，并且部分函数的均值并不是0，所以会出现如下图所示的情况。



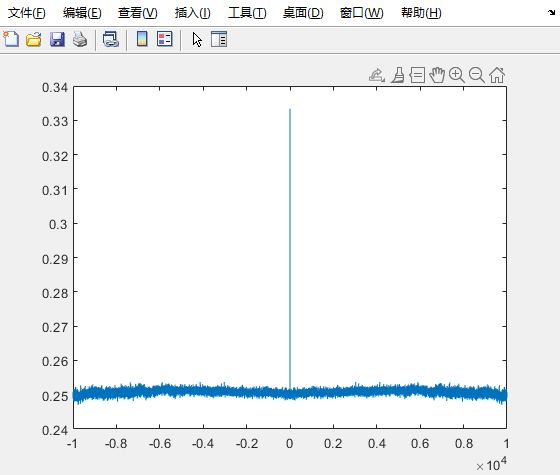
是因为各个偏移量能使用的数据数量是不同的，而且均值不为零，所以有些地方叠加的多，有些地方叠加的少，才导致这个样子。

使用unbiased参数，可以做一个对不同数量的样本的平均，这样我们就能得到一个较为合理的数据。



但是这里仍然存在一些问题，在两端由于数据量太小，会有很大的误差，导致两端会有翘起。

所以我们将原本的数据扩大，然后设置最大偏移，相当于只取中间的数据，就能得到与理论值相近的自相关函数。



五、实验内容实际应用

正文：

六、附录

仿真程序 1：exp1-1

1. clear all;
2. N = 20000;
3. maxlag = 10000;
4. x = poissrnd(1.0,N,1);
5. mu = mean(x)
6. myvar = var(x)
7. figure(1);
8. [n,edges] = histcounts(x);
9. n = n/N;
10. histogram(x,'Normalization','pdf'); % 使用直方图画离散随机变量的分布
11. % 这里之所以除了正态分布之外，没有在其他位置直接变为0，而是形成了一个近似三角形的一个区域；是因为各个偏移量能使用的数据数量是不同的，而且均值不为零，所以有些地方叠加的多，有些地方叠加的少，才导致这个样子。使用unbiased参数，可以做一个对不同数量的样本的平均，这样我们就能得到一个较为合理的数据。但是这样会有一个问题就是，在两端由于数据量太小，会有很大的误差，导致两端会有翘起。所以我们将原本的数据扩大，然后设置最大偏移，这样可以避免这样的问题。
12. [R\_x,lags]=xcorr(x,maxlag,"unbiased");
13. figure(2);
14. plot(lags,R\_x);
15. figure(3);
16. G\_X\_1 = fftshift(abs(fft(R\_x)));
17. % 对自相关函数进行傅里叶变换可以得到其功率谱密度，但是由于其自相关函数中的直流分量导致在0频率出有一个冲击函数
18. % 为了让图片更加清晰，我们将靠近0频的部分进行了去除
19. subplot(2,1,1);
20. plot(G\_X\_1);
21. xlim([0,2e4])
22. G\_X\_1(round((1+end)/2-10:round((1+end)/2+10))) = 0;
23. subplot(2,1,2);
24. plot(G\_X\_1);
25. xlim([0,2e4])

仿真程序 2：exp1-2

1. clear all;
2. clc
3. N = 200;
4. maxlag = 100;
5. x = normrnd(0,3,1,N);
6. mu = mean(x)
7. myvar = var(x)
8. figure(1)
9. [f,xi]=ksdensity(x); % 使用ksdensity估计概率密度
10. plot(xi,f);
11. %{
12. 之所以除了正态分布之外，自相关函数没有在其他位置直接变为0，而是形成了一个近似三角形的一个区域；
13. 是因为各个偏移量能使用的数据数量是不同的，而且均值不为零，所以中间叠加的多，两边叠加的少，才导致这个样子。
14. 使用unbiased参数，可以做一个对不同数量的样本的平均，这样我们就能得到一个较为合理的数据。
15. 但是这样会有一个问题就是，在两端由于数据量太小，会有很大的误差，导致两端会有翘起。
16. 所以我们将原本的数据扩大，然后设置最大偏移，这样相当于只取中间的数据。
17. %}
18. [R\_x,lags]=xcorr(x,maxlag,"unbiased");
19. figure(2);
20. plot(lags,R\_x);
21. G\_X\_1 = fftshift(abs(fft(R\_x)));
22. % 对自相关函数进行傅里叶变换可以得到其功率谱密度，但是由于其自相关函数中的直流分量导致在0频率出有一个冲击函数
23. % 会让整个图像变得不容易辨认，为了让图片更加清晰，我们将靠近0频的部分进行了去除
24. figure(3);
25. plot(G\_X\_1)

仿真程序 3：exp1-3

1. clear all
2. clc
3. N = 20000;
4. maxlag = 10000;
5. x = exprnd(0.5,N,1);
6. mu = mean(x)
7. myvar = var(x)
8. figure(1);
9. [f,xi]=ksdensity(x);
10. plot(xi,f);
11. [R\_x,lags]=xcorr(x,maxlag,"unbiased");
12. %{
13. 这里之所以除了正态分布之外，没有在其他位置直接变为0，而是形成了一个近似三角形的一个区域；
14. 是因为各个偏移量能使用的数据数量是不同的，而且均值不为零，所以有些地方叠加的多，有些地方叠加的少，才导致这个样子。
15. 使用unbiased参数，可以做一个对不同数量的样本的平均，这样我们就能得到一个较为合理的数据。
16. 但是这样会有一个问题就是，在两端由于数据量太小，会有很大的误差，导致两端会有翘起。
17. 所以我们将原本的数据扩大，然后设置最大偏移，这样相当于只取中间的数据。
18. %}
19. figure(2);
20. plot(lags,R\_x);
21. G\_X\_1 = fftshift(abs(fft(R\_x)));
22. % 对自相关函数进行傅里叶变换可以得到其功率谱密度，但是由于其自相关函数中的直流分量导致在0频率出有一个冲击函数
23. % 为了让图片更加清晰，我们将靠近0频的部分进行了去除
24. figure(3);
25. subplot(2,1,1)
26. plot(G\_X\_1)
27. xlim([0,2e4])
28. G\_X\_1(round((1+end)/2-10:round((1+end)/2+10))) = 0;
29. subplot(2,1,2)
30. plot(G\_X\_1);
31. xlim([0,2e4])

仿真程序 3：exp1-3

1. clear all
2. clc
3. close all
4. N = 20000;
5. maxlag = 10000;
6. x = rand(N,1);
7. mu = mean(x)
8. myvar = var(x)
9. figure(1);
10. [f,xi]=ksdensity(x);
11. plot(xi,f);
12. [R\_x,lags]=xcorr(x,maxlag,"unbiased");
13. %{
14. 这里之所以除了正态分布之外，没有在其他位置直接变为0，而是形成了一个近似三角形的一个区域；
15. 是因为各个偏移量能使用的数据数量是不同的，而且均值不为零，所以有些地方叠加的多，有些地方叠加的少，才导致这个样子。
16. 使用unbiased参数，可以做一个对不同数量的样本的平均，这样我们就能得到一个较为合理的数据。
17. 但是这样会有一个问题就是，在两端由于数据量太小，会有很大的误差，导致两端会有翘起。
18. 所以我们将原本的数据扩大，然后设置最大偏移，这样相当于只取中间的数据。
19. %}
20. figure(2);
21. plot(lags,R\_x);
22. G\_X\_1 = fftshift(abs(fft(R\_x)));
23. % 对自相关函数进行傅里叶变换可以得到其功率谱密度，但是由于其自相关函数中的直流分量导致在0频率出有一个冲击函数
24. % 为了让图片更加清晰，我们将靠近0频的部分进行了去除
25. figure(3);
26. subplot(2,1,1)
27. plot(G\_X\_1)
28. xlim([0,2e4])
29. G\_X\_1(round((1+end)/2-10:round((1+end)/2+10))) = 0;
30. subplot(2,1,2)
31. plot(G\_X\_1);
32. xlim([0,2e4])

# 基础实验二 利用蒙特卡洛方法估计随机事件的数字特征

一、实验内容简介

**1、实验目的：**

* 1. **通过实验加深对蒙特卡洛方法的认识**
  2. **了解蒙特卡洛方法在求解随机信号的数字特征中的应用**
  3. **熟悉matlab的相关操作**

**2、实验环境：**

1. 便携式计算机：宏碁传奇X
   1. CPU： Razen 7 5800U
   2. GPU： Nvidia GeForce RTX 3050
2. 系统环境：Windows10 系统家庭版
3. 软件环境：Matlab 2021a

**3、实验原理、实验内容及要求：**

1. 实验原理
   1. 在已知随机变量的概率密度分布函数的情况下，可以将其数字特征的求解转化为求解定积分的过程，并通过蒙特卡洛方法求得定积分的数值，即可得到对应的数字特征。

蒙特卡洛方法最常用的就是投点法，我们通过对一个正方形的矩形内随机投入点，并计算在对应函数下方的点占所有点的频率，当点足够多的时候频率就趋向于概率，再经过比例的缩放就能得到对应的定积分值。

* 1. 可以通过直接抽样法，模拟任意一个分布函数已知的随机分布。并通过统计的方法求得对应的数字特征。

1. 实验内容及要求
2. 生成一任意分布随机信号，并使用 MC 定积分方法求解随机变量数字特征，如：均值、方差、协方差、相关函数等。
3. 生成一任意分布随机信号，并使用 MC 模拟法求解其数字特征，如：均值、方差、协方差、相关函数等。

二、实验内容实现

**1、定积分法**

1. 选定一个概率密度分布函数已知的分布，这里我为了方便起见选择了使用正态分布，这样就能直接使用matlab的normpdf函数计算对应的概率密度。
2. 确定上下限：虽然正态分布的上下限是无穷，但是实际情况中由于边缘的可能性极小，我们可以使用一个有限值进行代替，相当于忽略了很小的一部分，经过测试，对整体的影响并不大。我生成了10000个正态分布的随机数，并统计其最小值与最大值，作为a与b。
3. 为了更加方便的使用蒙特卡洛方法进行定积分的运算，我编写了一个函数，通过输入对应方程，点数，以及上下限就能计算出对应的定积分数值。
4. 在这函数中，首先生成N个在[0,1]内的随机数对，并将其x放缩到[a,b]区间上，将其带入待求的方程中，并求其最大值与最小值M与m。将得到的方程值在再放缩至[0,1]区间内。将得到放缩之后的值与随机数对的y值进行比较，得到y<f的随机数对的数量，在除以随机数对的总数。对这一比例在根据a,b,m,M，进行放缩，就能得到最终的定积分值。
5. 对于求不同的随机变量，我们只需要写不同的积分函数即可。
   1. 对于其均值：f\_ex = @(x)x.\*p(x);其中，p(x)为normpdf(x,mu,sig)
   2. 对于其方差：f\_var = @(x)(x-mx).^2.\*p(x); 其中，p(x)为normpdf(x,mu,sig)
   3. 计算协方差，当t1≠t2时，我们可以得到

f\_Cx = @(x1,x2)(x1-mx).\*(x2-mx).\*normpdf(x1,mu,sig).\*normpdf(x2,mu,sig);

当t1 = t2时，我们有 x1 = x2，一定成立

f\_Cx = @(x)(x-mx).^2.\*p(x); （ 与f\_var相等 ）

* 1. 计算相关函数，当t1≠t2时，我们可以得到

f\_Rx = @(x1,x2)x1.\*x2.\*p(x1).\*p(x2);

当t1 = t2时，我们有 x1 = x2，一定成立

f\_Rx = @(x)x.^2.\*p(x);

1. 注意计算协方差以及相关函数时，函数需要计算一个二重定积分，我们可以将对应的随机数生成三维，继续使用蒙特卡洛方法进行计算。

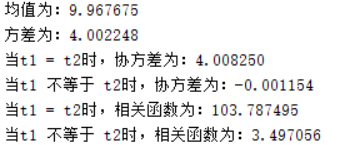
2、模拟法

1. 选定一个概率密度分布函数已知的分布，这里我为了方便起见选择了使用正态分布，这样就能直接使用matlab的normpdf函数计算对应的概率密度以及norminv计算其概率分布的逆函数。
2. 生成一个在区间[0,1]内的随机数列。
3. 将随机数列带入到norminv函数，得到随机变量X的抽样结果。
4. 利用得到的抽样结果使用统计方法进行数字特征的计算。
   1. 均值：直接使用mean()函数。
   2. 方差：直接使用var()函数。
   3. 相关函数：直接使用xcorr()函数。
   4. 协方差：使用相关函数再减去(E\_x)^2的方法求得。

三、实验仿真结论及分析

1、 定积分法

正态分布的参数我设置为mu = 10，sig = 2



可以看到，均值与方差基本与理论值相同。

当t1 = t2时，协方差等于方差，t1不等于t2时，基本为零，符合理论值。

当t1 = t2时，相关函数值较大，t1不等于t2时，相关函数很小，符合理论。

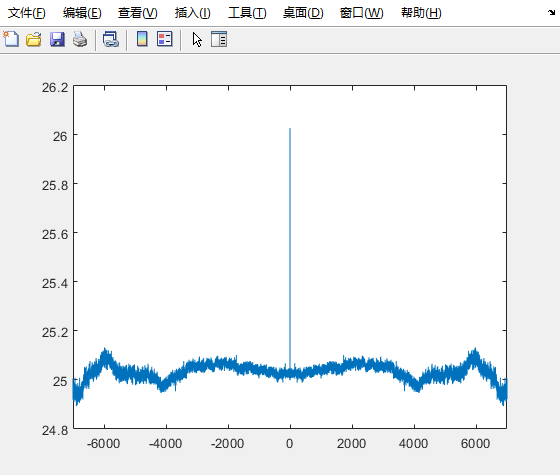
2、 模拟法

正态分布的参数我设置为 mu = 5，sig = 1

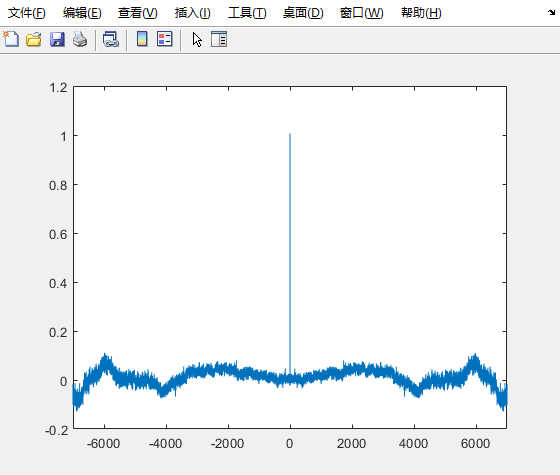


可以看到方差与均值基本与理论值相同

对于相关函数，同样有在实验一中遇到的问题，使用了同样的方法解决，得到的图像如下图所示。



对于协方差，由于是通过相关函数计算而来，其图像与相关函数基本相同，区别仅仅其在于减去了E\_x^2，这样使其均值趋向于0。



3、 影响估计值精度的主要因素有哪些？

1. 取点的数量。

2. 上下限的选择。

3. 概率密度本身的不同。

4、 不同方法求解随机变量数字特征的使用条件

1. 定积分法只能用于其概率密度函数已知的情况。

2. 模拟法用于概率分布函数已知的情况，并且对于不易采样的分布更加合适。

四、遇到的问题及解决的方法

对于相关函数以及协方差的计算有些疑问，在查阅各种资料以及进行了重新推导之后进行了分情况讨论，得到了较为合理的数值。

五、实验内容实际应用

正文：

六、附录

仿真程序 1：mcIntxy.m

1. function outputArg = mcIntxy(func,N,a,b)
2. xy = unifrnd(0,1,[2,N]);
3. x = a + (b-a).\*xy(1,:);
4. y = xy(2,:);
5. f = func(x);
6. m = min(f);
7. M = max(f);
8. f = (f - m)/(M-m);
9. n0 = sum(y<f);
10. outputArg = (M-m)\*(b-a)\*(n0/N)+m\*(b-a);
11. end

仿真程序 2：mcIntxyz.m

1. function outputArg = mcIntxyz(func,N,a,b,c,d)
2. xyz = unifrnd(0,1,[3,N]);
3. x = a + (b-a).\*xyz(1,:);
4. y = c + (d-c).\*xyz(2,:);
5. z = xyz(3,:);
6. f = func(x,y);
7. m = min(f);
8. M = max(f);
9. f = (f - m)/(M-m);
10. n0 = sum(z<f);
11. outputArg = (M-m)\*(b-a)\*(n0/N)+m\*(b-a);
12. end

仿真程序 3：exp2-1.m

1. clear all;
2. clc
3. N = 100000;
4. tmax = 0.5;
5. fs = N/tmax;
6. t = (1:N)/fs;
7. t1 = t;
8. t2 = t;
9. mu = 10;
10. sig = 2;
11. p = @(x)normpdf(x,mu,sig);
12. temp = normrnd(mu,sig,1,N);
13. a = min(temp)-3\*sig;
14. b = max(temp)+3\*sig;
15. % 计算均值
16. f\_ex = @(x)x.\*p(x);
17. mx = mcIntxy(f\_ex,N,a,b);
18. fprintf("均值为：%f\n",mx);
19. % 计算方差
20. f\_var = @(x)(x-mx).^2.\*p(x);
21. varx = mcIntxy(f\_var,N,a,b);
22. fprintf("方差为：%f\n",varx);
23. % 计算协方差，当t1 neq t2时，我们可以得到
24. f\_Cx = @(x1,x2)(x1-mx).\*(x2-mx).\*normpdf(x1,mu,sig).\*normpdf(x2,mu,sig);
25. Cx1 = mcIntxyz(f\_Cx,N,a,b,a,b);
26. % 当t1 = t2时，我们有 x1 = x2，一定成立
27. f\_Cx = @(x)(x-mx).^2.\*p(x); % 与f\_var相等
28. Cx2 = mcIntxy(f\_Cx,N,a,b);
29. fprintf("当t1 = t2时，协方差为：%f\n",Cx2);
30. fprintf("当t1 不等于 t2时，协方差为：%f\n",Cx1);
31. % 计算相关函数，当t1 neq t2时，我们可以得到
32. f\_Rx = @(x1,x2)x1.\*x2.\*p(x1).\*p(x2);
33. Rx1 = mcIntxyz(f\_Rx,N,a,b,a,b);
34. % 当t1 = t2时，我们有 x1 = x2，一定成立
35. f\_Rx = @(x)x.^2.\*p(x);
36. Rx2 = mcIntxy(f\_Rx,N,a,b);
37. fprintf("当t1 = t2时，相关函数为：%f\n",Rx2);
38. fprintf("当t1 不等于 t2时，相关函数为：%f\n",Rx1);

仿真程序 4：exp2-2.m

1. clear all;
2. clc
3. N = 10000;
4. sig = 1;
5. mu = 5;
6. p = unifrnd(0,1,1,N);
7. x = norminv(p,mu,sig);
8. E\_x = mean(x);
9. fprintf("均值为：%f\n",E\_x);
10. Var\_x = var(x);
11. fprintf("方差为：%f\n",Var\_x);
12. [R\_x,lag]=xcorr(x,'unbiased');
13. figure(1)
14. plot(lag,R\_x);
15. xlim([round(-N\*0.7),round(N\*0.7)])
16. C\_x = R\_x - E\_x\*E\_x;
17. figure(2)
18. plot(lag,C\_x);
19. xlim([round(-N\*0.7),round(N\*0.7)])