实验二 蒙特卡罗方法仿真

姓名　　学号　　班级

一、 实验目标

掌握蒙特卡罗方法的原理，蒙特卡罗方法在通信系统仿真中的应用。

二、 实验原理

1.蒙特卡洛的基本思想：

蒙特卡罗方法又称随机模拟方法，对研究的系统进行随机观察抽样,通过对样本值的统计分析，求得所研究系统的某些参数，它是在上世纪四十年代中期为了适应当时原子能事业的发展而发展起来，“曼哈顿计划”主持人之一、数学家：冯·诺伊曼用驰名世界的赌城—摩纳哥最大的城市 Monte Carlo—来命名这种方法。

优点：•Monte Carlo 方法及其程序结构简单。

•收敛速度与问题维数无关。

•Monte Carlo 方法的适用性强

缺点：•由于对基础风险因素的一些假设，使得存在模型风险。

•计算量大，准确性的提高速度慢。

•如果计算机模拟产生的是伪随机数，那么可能导致错误结果

2.蒙特卡洛的误差分析：

蒙特卡罗方法的误差 ε 定义为：

$$\varepsilon = \frac{u_\alpha \sigma}{\sqrt{n}}$$

①.蒙特卡罗方法的误差为概率误差，也即蒙特卡罗方法的收敛是概率意义下的收敛，虽然不能断言其误差不超过某个值，但能指出其误差以接近 1 的概率不超过某个界限。

②. 误差中的均方差 σ 是未知的，必须使用其估计值来代替，在计算所求量的同时，可计算出 σ。

减小误差的方式：

①.在 σ 固定的情况下，要把精度提高一个数量级，试验次数 n 需增加两个数量级。因此，单纯增大 n 不是一个有效的办法。

②.另一方面，如能减小估计的均方差 σ，比如降低一半，那误差就减小一半，这相当于 n 增大四倍的效果。

3.随机数的产生原理：

逆变换法(直接抽样方法)Inverse transform sampling

定理：设随机变量 U 服从(0,1)上的均匀分布，则 $X=F^{-1}(U)$的分布函数为 $F(x)$ 。

因此，要产生来自 $F(x)$的随机数，只要先产生来自 $U(0,1)$的随机数，然后计算 $F^{-1}(U)$ 即可。

$$\begin{cases}由U(0,1)抽取u，\\ 计算x=F^{-1}(u)\end{cases}$$

连续性分布函数的直接抽样方法对于分布函数的反函数容易实现的情况，使用起来是很方便的。但是对于以下几种情况，直接抽样法是不合适的：

•分布函数无法用解析形式表达，因而无法给出反函数的解析形式

•分布函数有解析形式，但是反函数的解析形式给不出来

•反函数有解析形式，但运算量很大

复合抽样方法

$$f(x)=\int f_2(x|y)\,dF_1(y)$$

如果 X 密度函数 $f(x)$难于抽样，而 X 关于 Y 的条件密度函数 $f_2(x|y)$以及 Y 的分布 $F_1(y)$均易于抽样，则 X 的随机数抽样：

首先从分布 $F^1(y)$ 中抽样 Y_{F1}，

然后再从密度函数 $f_2(x|\ Y_{F1})$中抽样确定 $X_{f2\ (x|YF)}$

筛选抽样 Acceptance-Rejection sampling

设 X 的密度函数 $f(x)$，且可将其表示成 $f(x)=ch(x)g(x)$，其中 $0<g(x)\le 1$，$c\ge 1$ 是常数，$h(.)$是一个密度函数,令 U 和 Y 分别服从 $U(0,1)$和 $h(y)$，则在 $U\le g(Y)$的条件下，Y 的条件密度为:

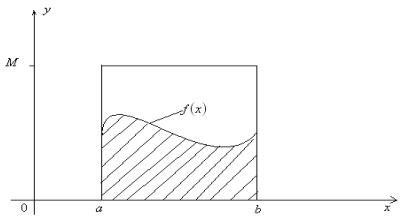
$$f_Y\left(y\middle|U\le g(Y)\right)=f(y)$$

依据上述定理，若 $h(.)$易于抽样，则 X 的抽样可如下进行：

$$\begin{cases}(1)由U(0,1)抽取u，\ 由h(y)抽取y,\\ (2)如果u\le g(y)，\ 则x=y，\ 停止，\\ (3)如果u>g(y)，\ 回到(1)。\end{cases}$$

4.定积分的 MC 计算：

随机投点法



方法简述：设 a，b 有限，$0<f(x)<M$，$\Omega=\{(x,y):a\le x\le b，\ 0\le y\le M\}$,并设$(X,Y)$是在$\Omega$上均匀分布的二维随机变量，其联合密度函数为：

$$\frac{1}{M(b-a)}I_{(a\le x\le b,0\le y\le M)}$$

则易见$\theta=\int_a^b f(x)\,dx$是Ω中 $y=f(x)$曲线下方的面积，假设我们向Ω中进行随机投点，则点落在 $y=f(x)$下方的概率 p， 若我们进行了 n 次投点，其中 n_0 次点落入 $y=f(x)$曲线下方，则用频率 n_0/n 来估计概率 p。即

$$\frac{n_0}{n}\approx p=\frac{\theta}{M(b-a)}$$

$$\begin{cases}\text{独立地产生}2n\text{个}U(0,1)\text{随机数，}u_i,\ v_i,\ i=1,\ldots,n,\\ \text{计算}x_i=a+u_i(b-a),\ y_i=Mv_i,\ \text{和}f(x_i)\\ \text{统计}f(x_i)\geq y_i\text{的个数}n_0\\ \text{用上式来估计}\theta\end{cases}$$

样本均值法

对积分$\theta=\int_a^b f(x)\,dx$，设 $g(x)$是(a, b)上的一个概率密度函数，改写

$$\theta=\int_a^b \frac{f(x)}{g(x)}g(x)dx=E\left[\frac{f(X)}{g(X)}\right]$$

其中，X 是服从 $g(x)$的随机变量．可见，积分可以表示为 X 的函数的期望。由矩法，若有 n 个来自 $g(x)$的观测值 $x_1,\ldots,x_n$，则可给出θ 的一个矩估计:

$$\theta=E[\frac{f(X)}{g(X)}]\approx\frac{1}{n}\sum_{i=1}^{n}\frac{f(x_i)}{g(x_i)}$$

特别地，若 a，b 有限，可取 $g(x)$ 为[a,b]上均匀分布．此时,设 $x_1,\ldots, x_n$ 是来自 $U(a,b)$的随机数，则θ 的一个估计为:

$$\hat{\theta}=\frac{1}{n}\sum_{i=1}^{n}\frac{f(x_i)}{g(x_i)}=\frac{b-a}{n}\sum_{i=1}^{n}f(x_i)$$

$$\begin{cases}\text{独立地产生}n\text{个}U(0,1)\text{随机数}u_1,\ldots,u_n,\\ \text{计算}x_i=a+(b-a)u_i,\ \text{和}f(x_i),\ i=1,\ldots,n,\\ \text{用上式来估计}\theta\end{cases}$$

三、 实验内容

第一部分：蒙特卡洛方法生成随机数

1、 用**逆变换法(直接抽样法)**产生指数分布

$$f(x)=\frac{1}{3}\exp\left(-\frac{1}{3}x\right)$$

的随机数。分别生成 100、1000、10000 个随机数，分别画出经验概率密度函数，并与概率密度函数曲线对比。

解：

指数分布为连续型分布，其一般形式如下：

$$f(x)=\lambda\cdot e^{-\lambda x},\quad x\geq 0$$

其分布函数为：

$$F(x)=1-e^{-\lambda x},\quad x\geq 0$$

则(1)由 $U(0,1)$抽取 u

(2)令 $F(x_F)=u$，得 $1-e^{-\lambda x_F}=u$，即 $-\lambda x_F=\ln(1-u)$

因为 1-u 也是(0,1)上均匀随机数，可将上式简化为：

$$x_F=-\ln u/\lambda$$

对应的算法流程为：

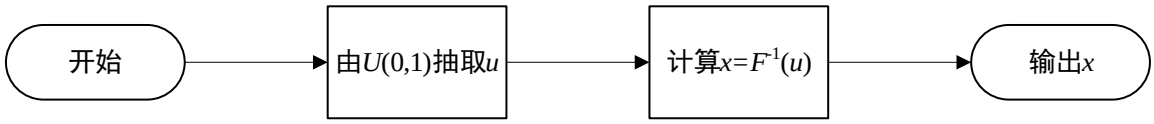


图1 逆变换法(直接抽样法)产生指数分布概率密度算法流程图

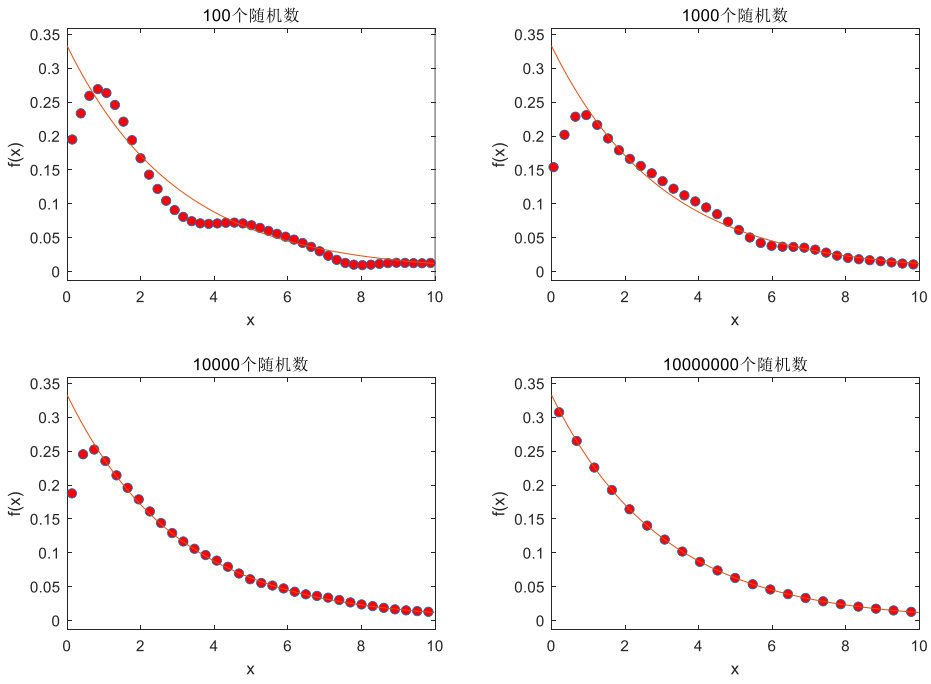


图2 逆变换法(直接抽样法)产生指数分布概率密度曲线对比图

如图2最终结果所示，随着随机数个数的不断增多，及实验次数的增加，最终越来越逼近于所求的概率密度分布。

2、设有X的密度函数为：

$$f(x)=0.6\times\frac{1}{2}\exp\left(-\frac{1}{2}x\right)+0.4\times 3\exp(-3x)$$

用**复合抽样法**，分别生成100、1000、10000个随机数，分别画出经验概率密度函数。

它相当于设Y为离散型随机变量，取1，2两个值，取1的概率为0.4，取2的概率为0.6。
当Y取1时，X的条件密度函数为: $f(x|Y=1)=3e^{-3x}$, $x>0$；
当Y取2时，X的条件密度函数为: $f(x|Y=2)=1/2e^{-x/2}$, $x>0$。
对应的算法流程为：

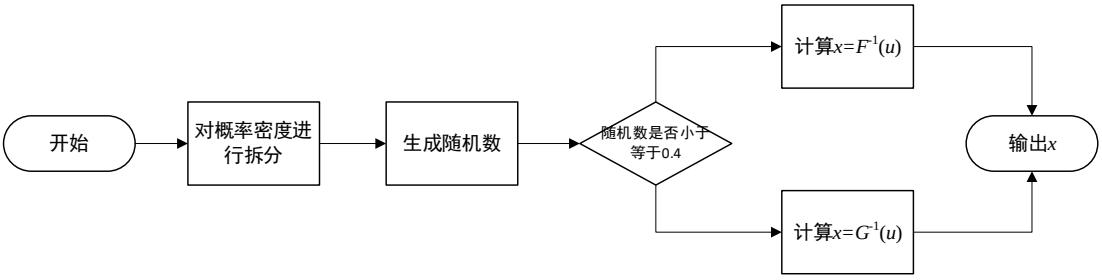


图3 复合抽样法产生指定分布概率密度曲线算法流程图

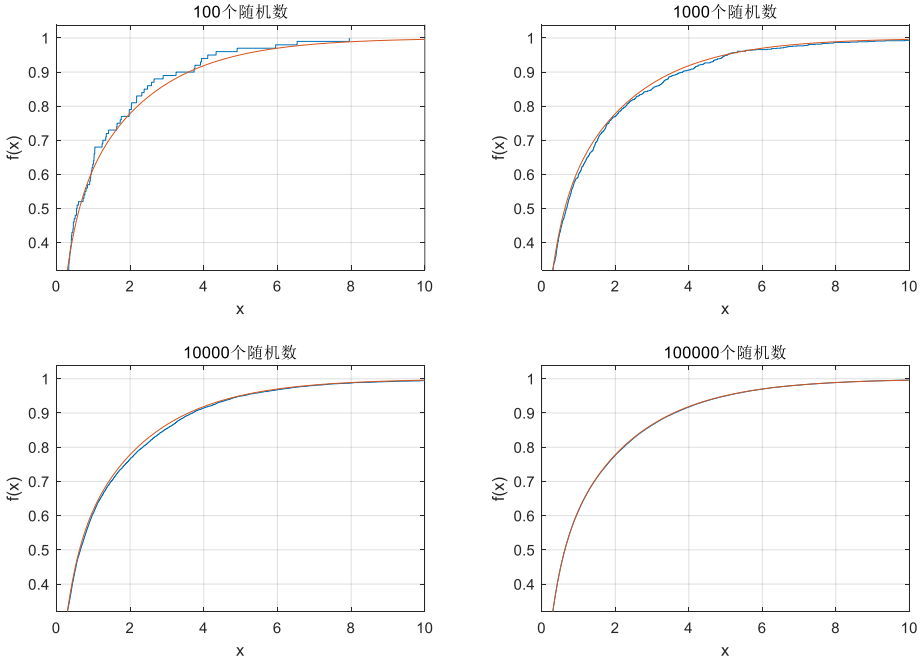


图4 复合抽样法产生指定分布概率密度曲线对比图

如图4最终结果所示，随着随机数个数的不断增多，及实验次数的增加，最终越来越逼近于所求的概率密度分布。

3、设球壳内半径为 $R_0=2$，外半径为 $R_1=5$，点到球心的距离 r 为，则 r 的分布密度函数为

$$f(r)=\begin{cases}\dfrac{3r^2}{R_1^3-R_0^3} & \text{当} R_0\le r\le R_1\\ 0 & \text{其它}\end{cases}$$

用**筛选抽样法**，分别生成100、1000、10000个随机数，分别画出散点图。

由题意可得取

$$h(r)=\frac{1}{R_1-R_2},\ g(r)=\frac{r^2}{R_0^2+R_0*R_1+R_1^2},\ c=3$$

算法流程图如下：

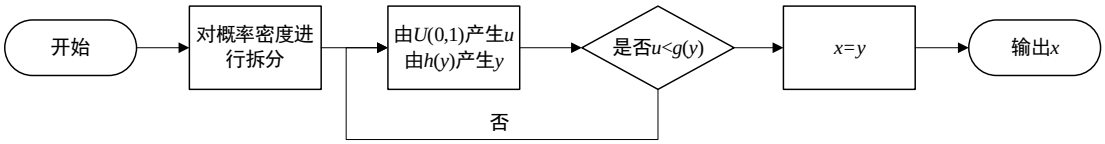


图5 筛选抽样法产生指定分布概率密度算法流程图

通过上述拆分后以图5所示完成筛选抽样最终得到结果如下图6所示。

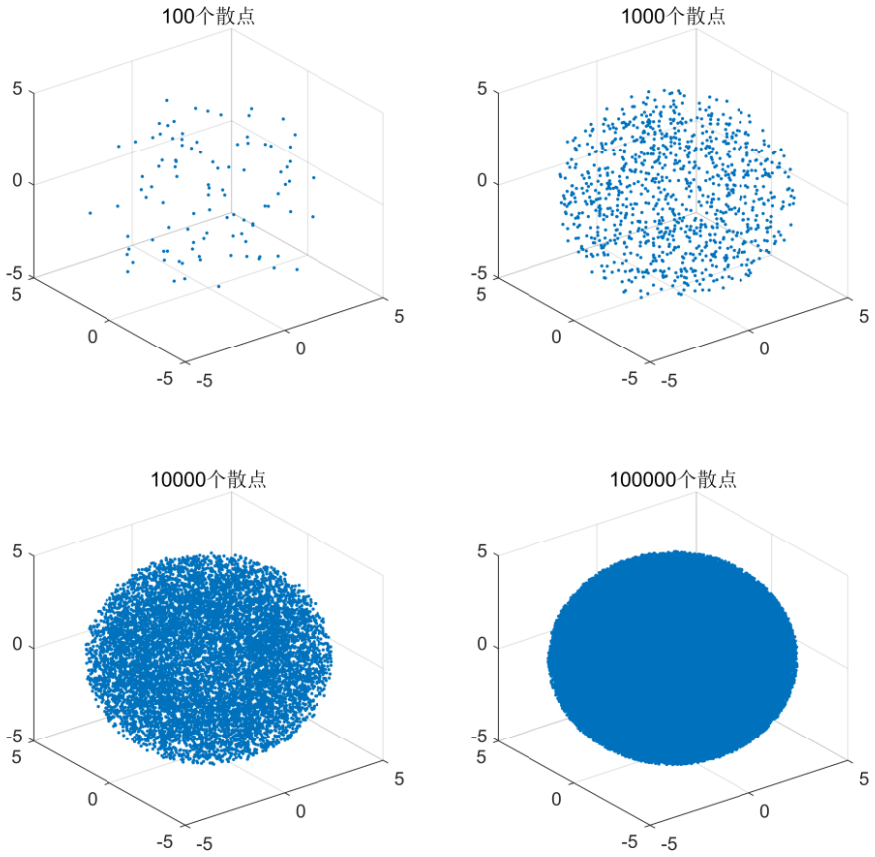


图6 筛选抽样法产生球壳分布散点图

如图6最终结果所示，随着随机数个数的不断增多，及实验次数的增加，最终越来越逼近于所求的概率密度分布。

第二部分：蒙特卡洛方法的应用

1、 计算 $\iint_D xy^2dxdy$ ，其中D为 $y=x-2$ 与 $y^2=x$ 所围的区域。

D的边界曲线交点为：(－1，1)，(4，2)，被积函数在求积区域内的最大值为16。积分值是三维体积，该三维图形位于立方体区域0≤ x ≤4，－1≤ y ≤2，0 ≤ z ≤16内，立方体区域的体积为192。

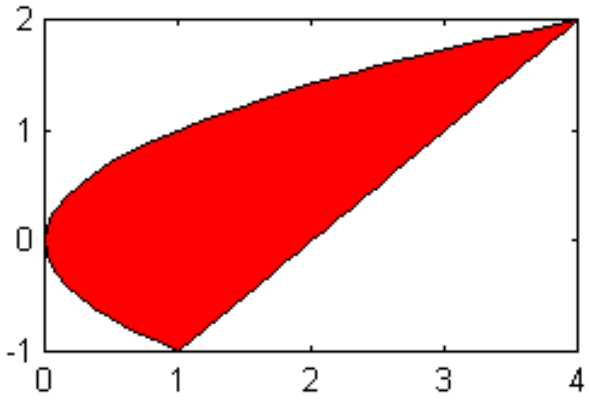


图7 积分区域平面图

最终仿真结果为：7.5902

```
V =
    7.5902
```

2、 设系统是由4个元件组成，每个元件的寿命服从期望为5的指数分布，每个元件是否正常工作相互独立。两系统的连接方式如下图所示，求系统寿命大于T=3的概率。（用随机投点法）

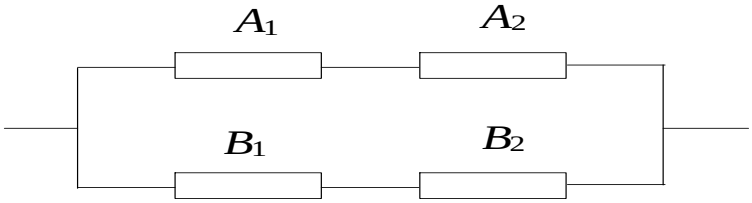


图8 电路系统示意图

理论分析：

$$P(S_1)=P((A_1A_2)\cup(B_1B_2))$$
$$=P(A_1A_2)+P(B_1B_2)-P(A_1A_2B_1B_2)$$
$$=2p^2-p^4=p^2(2-p^2)$$

最终通过仿真可得结果为：

```
Rguji =
    0.5124
```

3、 高斯白噪声信道下，采用相干检测的QPSK系统仿真。要求绘出系统原理框图、比特误

码率随SNR变化的曲线。

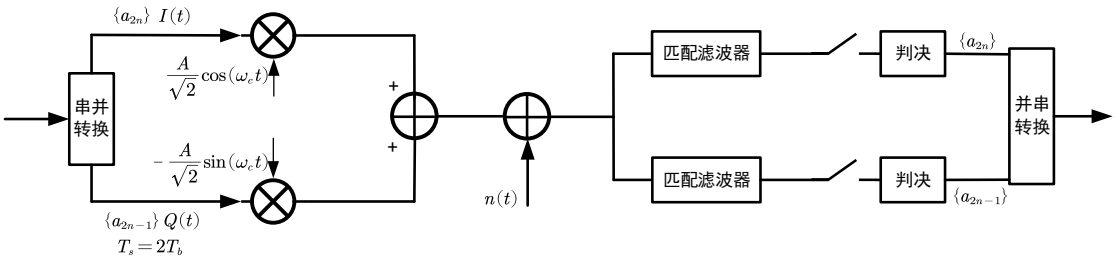


图9 QPSK系统原理框图

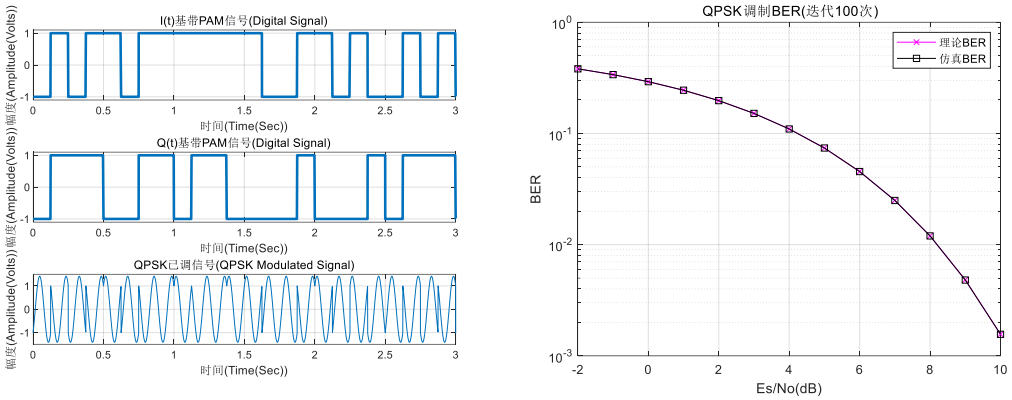


图10 QPSK仿真曲线图（左：QPSK信号 右：误码率仿真）

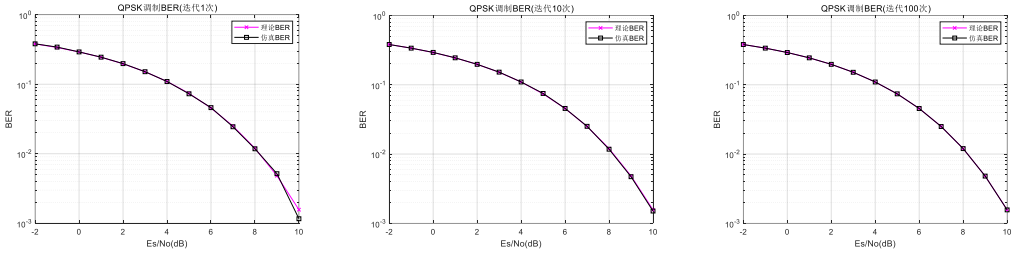


图11 不同迭代次数下QPSK仿真曲线图

如图11最终结果所示，随着随机数个数的不断增多，及实验次数的增加，最终越来越逼近于QPSK误比特率的理论值。

四、 结果分析

1、分析实验的误差与样本数的关系。

在 σ 固定的情况下，要把精度提高一个数量级，试验次数 n 需增加两个数量级。因此，单纯增大 n 不是一个有效的办法。

2、解析解与蒙特卡罗求解结果的对比分析。

由中心极限定理可知：

$$P\left(|\bar{X}_N - \mu| < \frac{u_\alpha \sigma}{\sqrt{n}}\right) \approx \Phi(u_\alpha) = 1 - \alpha$$

这表明，不等式 $|\bar{X}_n - \mu| < \frac{u_\alpha \sigma}{\sqrt{n}}$近似地以概率$1-\alpha$成立。

上式也表明，$\bar{X}_n$收敛到 μ 的阶为 $O(n^{-1/2})$。

通常，蒙特卡罗方法的误差 ε 定义为：

$$\varepsilon = \frac{u_\alpha \sigma}{\sqrt{n}}$$

随着试验次数增加，蒙特卡洛的仿真解越来越逼近理论解析解。

五、 实验心得

在进行蒙特卡罗模拟的 Matlab 实验中，我收获了很多关于随机模拟和数值计算的知识。通过编写代码和运行模拟，我学会了如何使用蒙特卡罗方法解决各种数学问题，并且更好地理解了随机变量和概率分布的概念。

在实验中，我遇到了一些问题，最常见的是在运行大规模模拟时内存不足的问题。我通过使用 Matlab 的向量化技术来提高代码的效率，并使用循环等方法减少内存占用。另外，我还学会了如何使用并行计算技术来加速模拟计算，从而更快地得出模拟结果。

另一个问题是如何确定模拟的精度和可靠性。我通过调整模拟参数，比如样本量和模拟次数，来评估模拟结果的精确性，并使用统计方法和可视化工具来验证模拟结果的可靠性。

总的来说，这个实验是一次非常有意义的学习经历，让我更深入地了解了蒙特卡罗方法的应用和实现，同时也提高了我的数学建模和编程技能。

六、 源代码

直接抽样法：

```
clear all;
clc;


%% ---------------- Parameters ---------------%%
Lambda = 1/3;

%% ----------------    指数分布   ---------------%%
figure;

% 100 个随机数
Randnum=-log(unifrnd(0,1,1,100))/Lambda;
[y,x]=ksdensity(Randnum);
subplot(2,2,1);
plot(x,y,'o','MarkerFaceColor','r')
hold on;
ezplot('1/3*exp(-1/3*x)',[0,10])
xlabel('x');
ylabel('f(x)');
```

```
title('100 个随机数')

% 1000 个随机数
Randnum=-log(unifrnd(0,1,1,1000))/Lambda;
[y,x]=ksdensity(Randnum);
subplot(2,2,2);
plot(x,y,'o','MarkerFaceColor','r')
hold on;
ezplot('1/3*exp(-1/3*x)',[0,10])
xlabel('x');
ylabel('f(x)');
title('1000 个随机数')

% 10000 个随机数
Randnum=-log(unifrnd(0,1,1,10000))/Lambda;
[y,x]=ksdensity(Randnum);
subplot(2,2,3);
plot(x,y,'o','MarkerFaceColor','r')
hold on;
ezplot('1/3*exp(-1/3*x)',[0,10])
xlabel('x');
ylabel('f(x)');
title('10000 个随机数')

% 100000 个随机数
Randnum=-log(unifrnd(0,1,1,10000000))/Lambda;
[y,x]=ksdensity(Randnum);
subplot(2,2,4);
plot(x,y,'o','MarkerFaceColor','r')
hold on;
ezplot('1/3*exp(-1/3*x)',[0,10])
xlabel('x');
ylabel('f(x)');
title('10000000 个随机数')
```

复合抽样法：

```
function xR = Composite_sampling(mm)
R=unifrnd(0,1,mm,1);R1=exprnd(1/3,mm,1);
R2=exprnd(2,mm,1);xR=zeros(mm,1);
for ii=1:mm
    if R(ii,1)<=0.4
       xR(ii,1)=R1(ii,1);
    else
```

```
            xR(ii,1)=R2(ii,1);
        end
    end
clc;
clear;


figure;
xR = Composite_sampling(100);
subplot(2,2,1);
cdfplot(xR);
hold on
ezplot('0.6*(1-exp(-0.5*x))+0.4*(1-exp(-3*x))',[0,10])
hold off
xlabel('x');
ylabel('f(x)');
title('100 个随机数')

xR = Composite_sampling(1000);
subplot(2,2,2);
cdfplot(xR);
hold on
ezplot('0.6*(1-exp(-0.5*x))+0.4*(1-exp(-3*x))',[0,10])
hold off
xlabel('x');
ylabel('f(x)');
title('1000 个随机数')

xR = Composite_sampling(10000);
subplot(2,2,3);
cdfplot(xR);
hold on
ezplot('0.6*(1-exp(-0.5*x))+0.4*(1-exp(-3*x))',[0,10])
hold off
xlabel('x');
ylabel('f(x)');
title('10000 个随机数')

xR = Composite_sampling(100000);
subplot(2,2,4);
cdfplot(xR);
hold on
ezplot('0.6*(1-exp(-0.5*x))+0.4*(1-exp(-3*x))',[0,10])
hold off
```

```
xlabel('x');
ylabel('f(x)');
title('100000 个随机数')
```

筛选抽样法：

```
function  [xRandnum,yRandnum,zRandnum]= Screening_sampling(R0,R1,
      mm)
xRandnum=zeros(1,mm);
yRandnum=zeros(1,mm);
zRandnum=zeros(1,mm);
ii=1;
while ii<mm
      Randnum1=unifrnd(-5,5);
      Randnum2=unifrnd(-5,5);
      Randnum3=unifrnd(-5,5);
      s=Randnum1^2+Randnum2^2+Randnum3^2;
      if  (s<=R1^2)&&(s>=R0^2)
       xRandnum(1,ii)=Randnum1;
       yRandnum(1,ii)=Randnum2;
       zRandnum(1,ii)=Randnum3;
       ii=ii+1;
     end
end
clc;
clear;

R0=2;
R1=5;

[xRandnum,yRandnum,zRandnum] = Screening_sampling(R0,R1,100);
subplot(2,2,1);
scatter3(xRandnum,yRandnum,zRandnum,'.');
xlim([-5,5]);
title('100 个散点');

[xRandnum,yRandnum,zRandnum] = Screening_sampling(R0,R1,1000);
subplot(2,2,2);
scatter3(xRandnum,yRandnum,zRandnum,'.');
xlim([-5,5]);
title('1000 个散点');

[xRandnum,yRandnum,zRandnum] = Screening_sampling(R0,R1,10000);
subplot(2,2,3);
```

```
scatter3(xRandnum,yRandnum,zRandnum,'.');
xlim([-5,5]);
title('10000 个散点');

[xRandnum,yRandnum,zRandnum] = Screening_sampling(R0,R1,100000);
subplot(2,2,4);
scatter3(xRandnum,yRandnum,zRandnum,'.');
xlim([-5,5]);
title('100000 个散点');
```

求定积分：

```
          data=rand(100000000,3);
           x=4*data(:,1);
           y=-1+3*data(:,2);
           z=16*data(:,3);
           II=find(x>=y.^2&x<=y+2&z<=x.*(y.^2));
           M=length(II);
           V=192*M/100000000
```

可靠性分析：

```
function Rguji=kekao_fenxi(t,thetaa1,thetaa2,thetab1,thetab2,mm)
%t 是要求系统生存的寿命%thetaa1 是元件 A1 的数学期望
%thetaa2 是元件 A2 的数学期望%thetab1 是元件 B1 的数学期望
%thetab2 是元件 B2 的数学期望%mm 是随机实验次数
frq=0;randnuma1 = exprnd(thetaa1,1,mm);
randnuma2 = exprnd(thetaa2,1,mm);
randnumb1 = exprnd(thetab1,1,mm);
randnumb2 = exprnd(thetab2,1,mm);
for ii=1:mm
    if (randnuma1(1,ii)>t)&(randnuma2(1,ii)>t)        pass1=1;
    else         pass1=0;
    end
    if (randnumb1(1,ii)>t)&(randnumb2(1,ii)>t)         pass2=1;
    else        pass2=0;
    end
    if (pass1+pass2)>=1        frq=frq+1;
    end
end,Rguji=frq/mm

clc;
clear;

```

```
Rguji=kekao_fenxi(3,5,5,5,5,1000000);
```

QPSK 仿真：

```
%% <<<<<<<<<<<<<<<<<< QPSK Modulation and Demodulation >>>>>>>>>>
      >>>>>>>>>>
clc;
clear all;
close all;

%% ******************* 初始化 ******************** %%
M = 4; %四进制
fc = 8; %载波频率
Rb = fc; %码元速率
T = 1 / Rb; % 一个码元传送的时间
dt = 1/2000; %一个码元的抽样时间
fs = 1/dt; %采样频率
nfft=1024; %采样的点数
Nbit = 100000;%10^5 个二进制比特
N = Nbit/log2(M);
t=0:dt:N*T-dt;% 所有码元的时间
ta=0:dt:T-dt; % 一个码元时间向量
%此时，一个码元序列被抽样 T/dt 次

A = 1;%振幅等于 1
SNR_dB = -2:10; %信噪比——SNR dB 值
SNR = 10.^(SNR_dB./10); %信噪比——SNR
Es= A^2*T/2; % QPSK 频带信号对应的维度的每符号能量
N0 = Es./SNR ; %QPSK 频带信号对应的噪声功率
theo_err_prb=[];
iter = 100; %迭代次数
error_rate = zeros(iter,length(SNR_dB));

%% 随机比特序列、以及 PAM 生成 %%
a = 2*(round(rand(1,Nbit))-1/2); %二进制比特随机序列
for i = 1:N
    bI(1,i) = a(1,2*i);
    bq(1,i) = a(1,2*i - 1);
end

gtl = (ones(1,T/dt))'*bI;  %生成 T/dt 列的二进制比特序列
g_PAM_I = gtl(:)';
gtq = (ones(1,T/dt))'*bq;  %生成 T/dt 列的二进制比特序列
g_PAM_q = gtq(:)';
```

```
t1 = length(t);

%% 载波生成以及 QPSK 频带调制 %%
f_I = A*cos(2*pi*fc*t); %生成载波信号
f_q = A*sin(2*pi*fc*t); %生成载波信号
qpsk = g_PAM_I.*f_I  - g_PAM_q.*f_q; %QPSK 频带调制信号的生成

%% QPSK 频带已调信号功率谱 %%
% window=boxcar(length(psk));%使用矩形窗
% nw=4;                                          %为时间带宽积，缺省值为 4
% [Pxx_PAM,ff_PAM]=pmtm(g_PAM,nw,nfft,fs); %用多窗口法(NW=4)估计功率谱
% [Pxx_PSK,ff_PSK]=pmtm(psk,nw,nfft,fs); %用多窗口法(NW=4)估计功率谱
% P_PAM= [fliplr(Pxx_PAM) Pxx_PAM];
% ff_PAM = [-fliplr(ff_PAM) ff_PAM];
% P_PSK= [fliplr(Pxx_PSK) Pxx_PSK];
% ff_PSK = [-fliplr(ff_PSK) ff_PSK];

%% 时域信号绘图 %%
figure(1);
subplot(3,1,1);
plot(t,g_PAM_I,'LineWidth',2);
title('I(t)基带 PAM 信号(Digital Signal)');
xlabel('时间(Time(Sec))');
ylabel('幅度(Amplitude(Volts))');
grid on;
axis([0,3,-1.1,1.1]);
subplot(3,1,2);
plot(t,g_PAM_q,'LineWidth',2);
title('Q(t)基带 PAM 信号(Digital Signal)');
xlabel('时间(Time(Sec))');
ylabel('幅度(Amplitude(Volts))');
grid on;
axis([0,3,-1.1,1.1]);
subplot(3,1,3);
plot(t,qpsk);
title('QPSK 已调信号(QPSK Modulated Signal)');
xlabel('时间(Time(Sec))');
ylabel('幅度(Amplitude(Volts))');
grid on;
axis([0,3,-1.5,1.5]);

%% 功率谱绘图
```

```
% figure(2);
% subplot(2,1,1);
% plot(ff_PAM,P_PAM);
% title('基带 PAM 信号的功率谱');
% xlabel('频率(Hz)');
% ylabel('幅度');
% grid on;
% xlim([-200,200]);
% subplot(2,1,2);
% plot(ff_PSK,P_PSK);
% title('频带 PSK 信号的功率谱');
% xlabel('频率(Hz)');
% ylabel('幅度');
% grid on;
% xlim([-200,200]);

%% QPSK 信号转化为信号空间中 %%
QPSK_I = g_PAM_I./ sqrt(1/Es);
QPSK_q = g_PAM_q./ sqrt(1/Es);
QPSK_signal = QPSK_I + 1j * QPSK_q;
s = length(ta);
for n = s:s:length(QPSK_signal)
    qpsk_signal(1,n/s) = QPSK_signal(1,n);
end

%% AWGN 信道传输 %%
s00 = 1/sqrt(1/Es)*(-1-1j);
s01 = 1/sqrt(1/Es)*(-1+1j);
s10 = 1/sqrt(1/Es)*(1-1j);
s11 = 1/sqrt(1/Es)*(1+1j);

for i = 1:iter
    for p = 1:length(SNR_dB)
        noise1 = sqrt(N0(p)) * randn(1,length(qpsk_signal)); %产生 QPSK 频带信号对应的加性高斯白噪声
        noise2 = sqrt(N0(p)) * randn(1,length(qpsk_signal)); %产生 QPSK 频带信号对应的加性高斯白噪声
        out_qpsk = qpsk_signal +(noise1 + 1j*noise2);
        for k =1:length(out_qpsk)
            d00 = abs(out_qpsk(k)-s00);
            d01 = abs(out_qpsk(k)-s01);
            d10 = abs(out_qpsk(k)-s10);
            d11 = abs(out_qpsk(k)-s11);
            if((d00<d01)&&(d00<d10)&&(d00<d11))   %最小距离判决
```

```
                detect(1,k) = -1/sqrt(1/Es)-1j/sqrt(1/Es);
            elseif((d01<d00)&&(d01<d10)&&(d01<d11))
                detect(1,k) = -1/sqrt(1/Es)+1j/sqrt(1/Es);
            elseif((d10<d00)&&(d10<d01)&&(d10<d11))
                detect(1,k) = 1/sqrt(1/Es)-1j/sqrt(1/Es);
            else
                detect(1,k) = 1/sqrt(1/Es)+1j/sqrt(1/Es);
            end
        end
        theo_err_prb(p)=2*qfunc(sqrt(SNR(p)))-qfunc(sqrt(SNR(p)))
      .*qfunc(sqrt(SNR(p)));
        %% erro pro
        simulated_err_prb(p) = 1 - sum(detect == qpsk_signal)/N;
        if p == 8
            r_bits_5dB = out_qpsk;
        elseif p == 13
            r_bits_10dB = out_qpsk;
        end
    end
    error_rate(i,:) = simulated_err_prb;
end
err_rate = mean(error_rate,1);
figure;
semilogy(SNR_dB,theo_err_prb,'M-X',SNR_dB,err_rate,'k-s');
grid on;
xlabel('Es/No(dB)');
ylabel('误码率');
title('QPSK 调制误码率');
legend('理论误码率','仿真误码率');
scatterplot(r_bits_5dB);
title('Receiving signal constellation diagram under 5dB');
line([0 0],[-3 3],'Color','r');
line([-3 3],[0 0],'Color','r');
grid on;
scatterplot(r_bits_10dB);
title('Receiving signal constellation diagram under 10dB');
line([0 0],[-3 3],'Color','r');
line([-3 3],[0 0],'Color','r');
grid on;
scatterplot(detect);
title('Decoded signal constellation');
line([0 0],[-3 3],'Color','r');
line([-3 3],[0 0],'Color','r');
grid on;
```