通信系统仿真
第2章 蒙特卡罗方法

何晨光
哈尔滨工业大学
电子与信息工程学院

1 蒙特卡洛方法简史

2 基本概念

3 在通信系统中的应用

4 蒙特卡罗积分

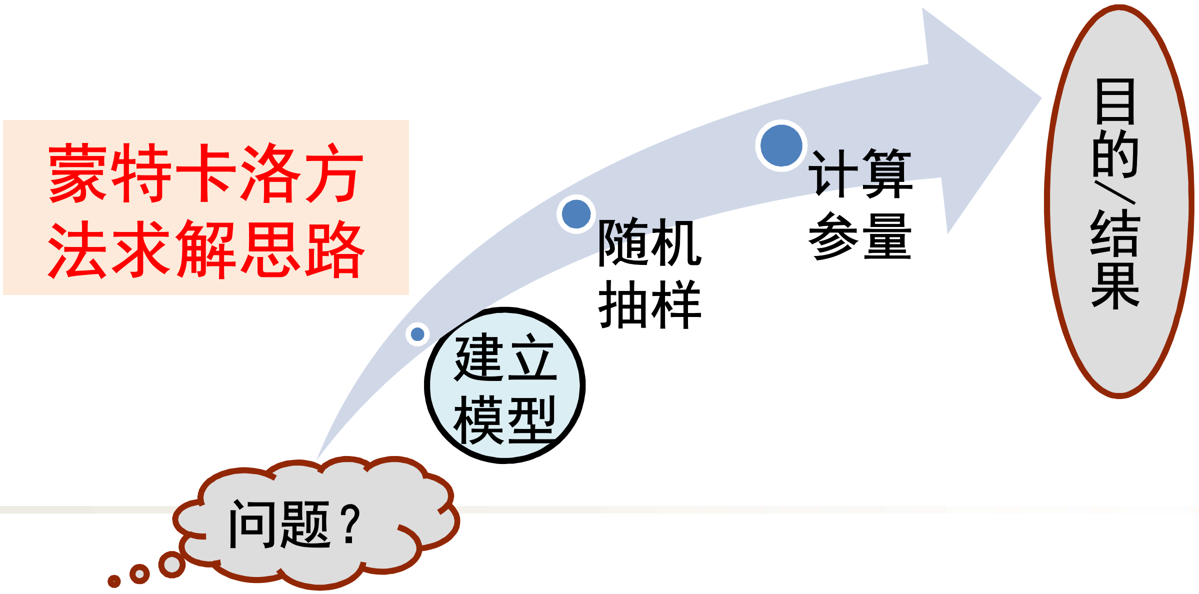
5 繁琐通信系统中的应用

6 半解析方法

1 蒙特卡洛方法简史

Monte Carlo方法，又称随机模拟方法或统计试验方法，属于计算数学的一个分支。

它是以概率统计理论为基础，依据大数定律（样本均值代替总体均值），利用电子计算机数字模拟技术，通过不断产生随机序列来模拟事件的过程，来解决一些很难直接用数学运算求解或其他方法不能解决的复杂问题的一种近似算法，它既可模拟随机事件，也可模拟确定性事件。



1 蒙特卡洛方法简史

- 蒙特卡洛(Monte Carlo)名字由来

Monte Carlo 是地中海之滨摩纳哥(Monaco)公国最大的城市，也是世界三大赌城之一（另两个是美国拉斯维加斯和中国澳门）。

1 蒙特卡洛方法简史

□ 蒙特卡洛(Monte Carlo)名字由来

摩纳哥(Monaco)位于欧洲的一个城邦国家，是欧洲四个公国之一（另三个是列支敦士登、卢森堡、安道尔），也是世界第二小的国家（面积最小的是梵蒂冈）。

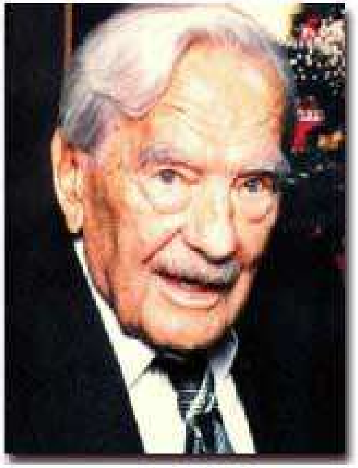
1 蒙特卡洛方法简史

□ 蒙特卡洛(Monte Carlo)方法的提出

二战期间，为解决原子弹研制中裂变物质中子随机扩散问题，美国科学家 **V. Neumann**、**S. Ulam** 和 **N. Metropolis** 提出一种随机模拟方法，出于保密，给这一方法起了个代号——Monte Carlo。用赌城名字作为该方法名称，既反映了方法的内涵，又为方法蒙上了一层神秘色彩。

Von Neumann
(1903-1957)

Stanislaw Ulam
(1909-1984)



Nicholas Metropolis
(1915-1999)

曼哈顿计划

1 蒙特卡洛方法简史

□ 蒙特卡洛(Monte Carlo)方法的应用

计算物理学：粒子运输、量子热力学、空气动力学、核工程等

数学：多重积分、解微分方程、非线性方程组求解等

工程领域：电子信息技术、真空技术、力学、激光技术等

经济学领域：期权定价、项目管理、投资风险决策等

其他领域：化学、医学，生物，生产管理、系统科学、公用事业等方面。随着科学技术的发展，其应用范围将更加广泛。

2 基本概念

基本思想是:

- 首先建立一个概率模型或随机过程，使它的参数等于问题的解。即确定随机试验和一个感兴趣的事件。
- 然后通过对模型或过程的观察或抽样试验来计算所求随机参数的统计特征;
- 最后给出所求解的近似值，解的精度可用估计值的标准误差来表示。

2.1 相对频率

蒙特卡罗是基于概率的相对频率解释的。

- 由基本的概率论可知，随机试验中实验结果无法准确预测，而只能用统计的方法加以描述。

- 首先确定随机试验和一个感兴趣的事件A。

一个最基本的随机试验就是掷硬币，其中实验结果有正，反两个。在掷硬币前是无法知道会出现什么结果的，但是，如果硬币是公平的（无偏的），则正反每一种结果出现的概率会想等，并且是互相独立的。

- 随机试验的性能决定了试验结果。

2.1 相对频率

随机试验中事件可以是一个结果或者几个结果的集合。

例：以数字通信系统为例，确定系统的BER。

- 随机试验定义为发送一个二进制数1，接收机输出端的结果为对所发送二进制符号的估计，是二进制数0或者1。
- 感兴趣的事件是发送1的过程中产生的差错。
- 系统的BER，就是在发送1的条件下接收到0这一事件的概率。

2.1 相对频率

在定义了随机试验和感兴趣的事件之后，开始进行大量的随机试验。

- 试验次数为N，以N_A表示事件A发生的次数。将事件A发生的概率近似为相对概率，定义为N_A/N。

- 事件A的概率可以通过重复无限次随机试验来求得，即

$$P_r(A) = \lim_{N \to \infty} \frac{N_A}{N}$$

- N是总的发送比特数或符号数（系统实际发送的或仿真的）
- N_A是差错发生的次数（测量出来的或仿真得到的）

2.1 相对频率

- 在蒙特卡罗仿真中，显然必然有 $N < \infty$ ，数值N_A/N就是Pr(A)估计器，该估计器记作

$$\hat{\Pr}(\mathrm{A})$$

- 由于试验的随机性，在试验次数N有限的情况下，N_A是随机变量，因此 $\hat{\Pr}(\mathrm{A})$ 也是随机变量。

- 该随机变量的统计性质决定了估计器的精度，因而也决定了仿真的质量。

2.2 蒙特卡罗估计器的性质

- 无偏性 $E\{\hat{A}\} = A$，即在平均意义下可以得到正确的结果

- 一致性 $N \to \infty, \quad \sigma_{\hat{A}}^2 \to 0$

- 解释：如果估计值是无偏的并具有小的方差，则估计器所产生的估计值会聚集在待估计参数真值的周围，且有较小的散布范围。除非所研究的事件是统计独立的，用解析的方法确定蒙特卡洛估计器的方差是非常困难的，但是估计器的方差会随着仿真时间（实验重复次数）的增加而减少。我们把满足这一性质的估计器称为一致的。

- 无偏和一致估计器，设误差为 $e = A - \hat{A}$

则有 $E[e]=0$

$$N \to \infty, \quad \sigma_e^2 \to 0$$

2.3 蒙特卡罗估计

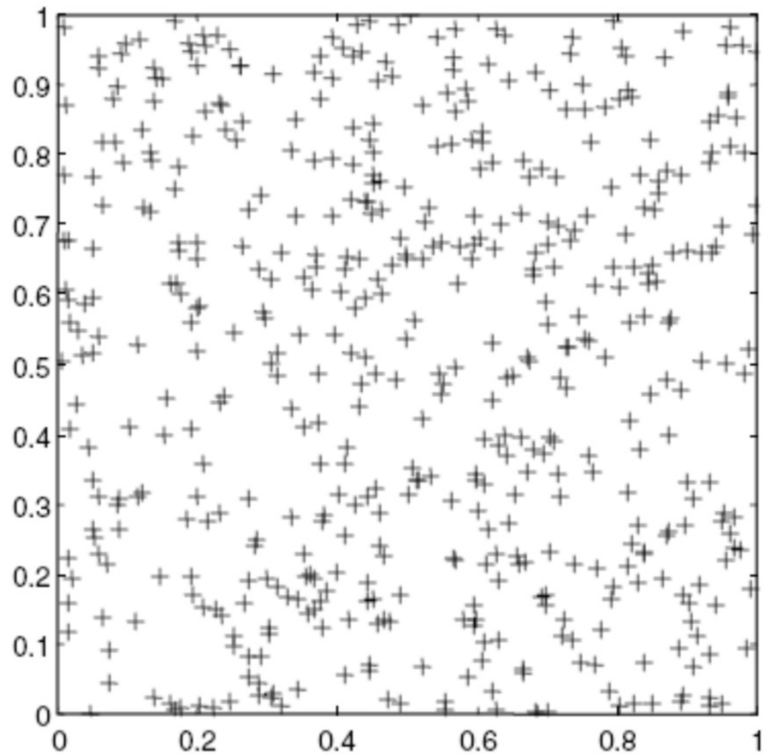
例：确定一个形状不规则的区域的面积

- 假设条件：待估计面积的区域完全包含在一个面积已知的方框中
- 随机试验：在包围方框中随机抽取采样，事件A定义为采样点落在面积待定的区域内
- 试验过程：利用两个均匀随机数在面积已知的包围区域内投点

2.3 蒙特卡罗估计

- 产生500个均匀分布的随机样点

```
X = rand(1,500);
Y = rand(1,500);
plot(X, Y, ’k+’)
axis square
```

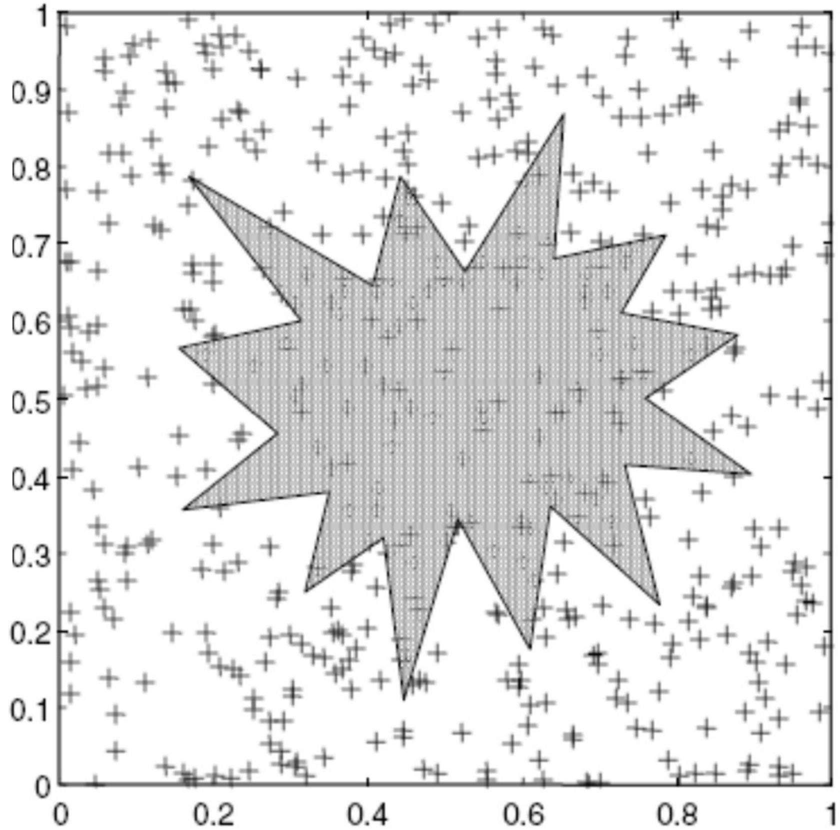


2.3 蒙特卡罗估计

- N_{box}和$N_{sunburst}$ 为落在方框中和爆炸形(sunburst)区域的面积。
- 右图为采样点在方框中是均匀分布的，$A_{sunburst}/A_{box}$近似为采样点落在爆炸区域内的数目与落在方框内的数目之比。

$$\frac{A_{sunburst}}{A_{box}} \approx \frac{N_{sunburst}}{N_{box}}$$

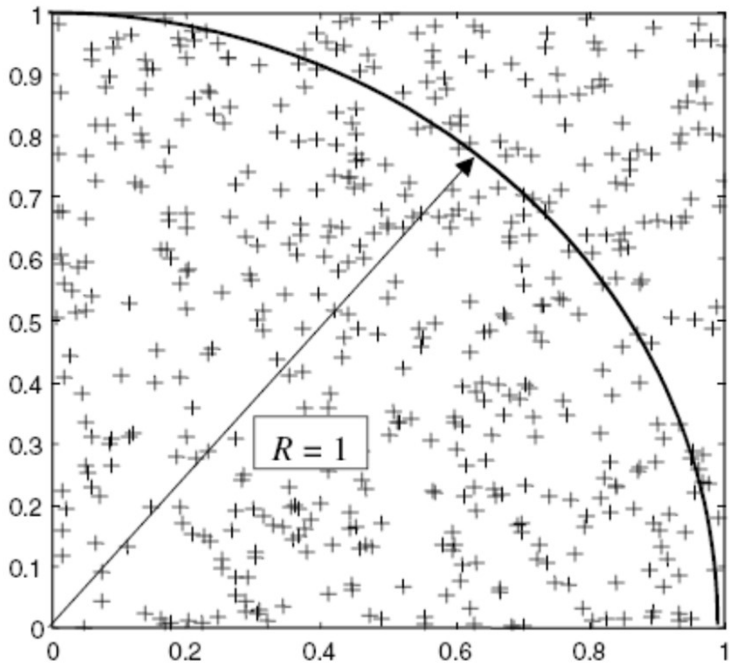
$$A_{sunburst} \approx A_{box}\frac{N_{sunburst}}{N_{box}}$$



- 在采样点为均匀分布这一条件下，随着采样点数的增加，近似精度会不断提高

2.4 π值的蒙特卡罗估计

用具有单位面积的正方形包围单位圆的第一象限



- 对π值的估计，只要在包围方框中撒上均匀分布的采样点，在对落在内切圆中的采样点计数，然后按照上面的公式对π进行估计。

2.4 π值的蒙特卡罗估计

包围区域面积 $A_{box}=1$

¼圆面积 $A_{pie\_slice}=[\pi R^2/4]_{R=1}=\pi/4$

面积比 $A_{box}/A_{pie\_slice}=\pi/4$

假设采样点是均匀分布的，则N_{pie_slice}和N_{box}之比构成了的无偏估计，所以有

采样点比 $N_{pie\_slice}/N_{box}\approx A_{pie\_slice}/A_{box}=\pi/4$

π的估计器 $$\hat{\pi}=\frac{4N_{pie\_slice}}{N_{box}}$$

2.4 π值的蒙特卡罗估计

```
m=input('Enter M, the number of experiments > ');
n=input('Enter N, the number of trials per experiment > ');
z=zeros(1,m);
data = zeros(n,m);
for j=1:m
    x=rand(1,n);
    y=rand(1,n);
    k=0;
    for i=1:n
        if x(i)^2+y(i)^2 <= 1        % Fall inside pie slice?
            k=k+1;
        end
        data(i,j) = 4*(k/i);         % jth estimate of pi
    end
    z(j) = data(n,j);                % Store data
end
plot(data,'k')                       % Plot curves
xlabel('Number of Trials')
ylabel('Estimate of pi')
% End of script file.
```

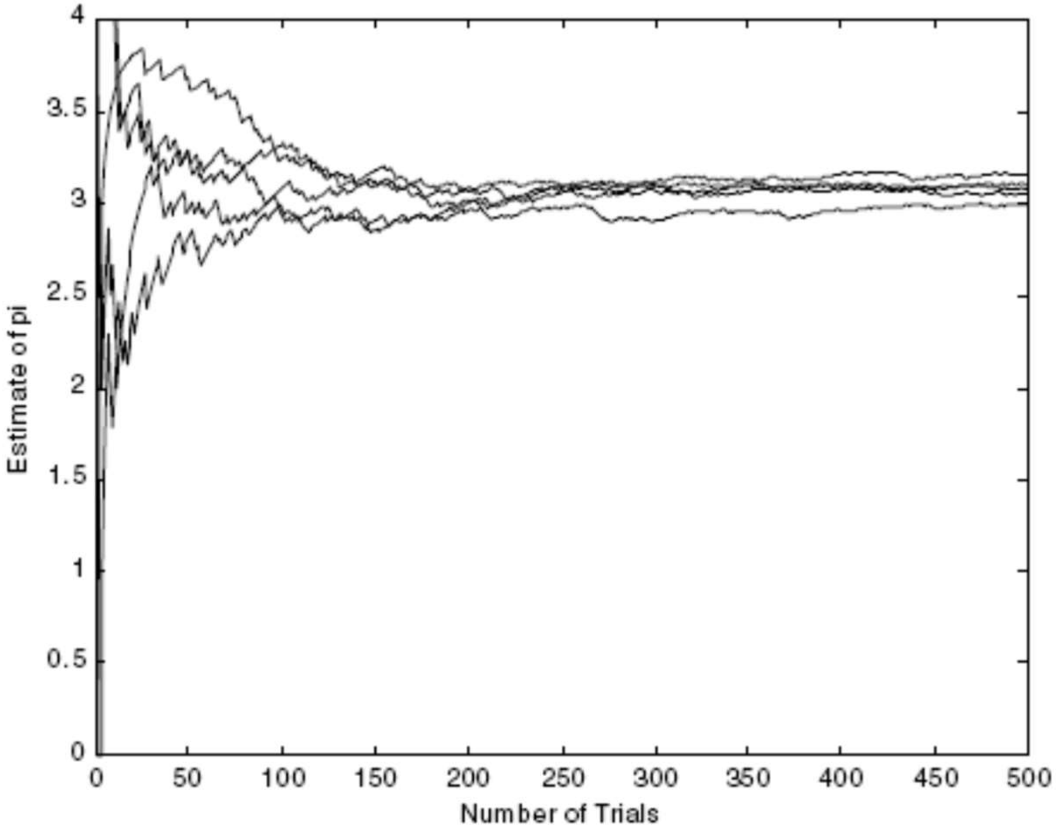
（代码见c201.m）

2.4 π值的蒙特卡罗估计

- 试验结果：5个π值估计，每个基于500次试验

$$\hat{\pi} = \begin{bmatrix} 3.0960 & 3.0720 & 2.9920 & 3.1600 & 3.0480 \end{bmatrix}$$

- 取平均得 $\hat{\pi} = 3.0736$



2.4 π值的蒙特卡罗估计

蒙特卡罗仿真的特点:

- 一个测试条件和两个计数器。
- 随机试验每进行1次，第一个计数器就增加1，如果测试条件每满足一次，第二个计数器就增加1。

```
for i=1:n
    if x(i)^2+y(i)^2 <= 1
        k=k+1;
```

2.4 π值的蒙特卡罗估计

对于数字通信系统的仿真

- 仿真的目的是估计误比特率
- 测试条件是在发送给定的比特或者数据符号时是否发生了差错。
- 每当发送了一个比特或者数据符号，随机试验每进行1次，第一个计数器就增加1。
- 每当观测到一次差错，第二个计数器就增加1。

2.4 π值的蒙特卡罗估计

小实验：改变C201.m中的M和N值，对结果会有什么影响?

3 在通信系统中的应用
—AWGN信道

- 方法：让N个符号通过系统（系统的计算机仿真模型），并计算发送差错的个数Ne。如果在N次的符号发送中有Ne次差错，则符号差错概率为

$$\hat{P}_E = \frac{N_e}{N}$$

- 问题：有偏或无偏？一致估计？

- 简化：AWGN信道假设，所产生的差错相互独立，则N次符号发送中出现的差错次数Ne可以用二项分布来描述。

3.1 二项式分布

- 确定$\hat{P}_E$的统计特性。首先确定Ne的均值和方差。
- 相互独立的差错事件，N次符号发送中有Ne次差错的概率服从二项式分布

$$P_N(N_e)=\binom{N}{N_e}P_E^{N_e}(1-P_E)^{N-N_e}$$

式中 $\binom{N}{k}=\dfrac{N!}{k!(N-k)!}$ 是二项式分布系数，

P_E是单次发送时的差错概率

3.1 二项式分布

- 服从二项式分布的随机变量的均值和方差

$$E\{N_e\} = NP_E$$

$$\sigma_{N_e}^2 = NP_E(1-P_E)$$

- 差错概率的蒙特卡罗估计器的均值和方差

$$E\{\hat{P}_E\} = \frac{E\{N_e\}}{N} = \frac{NP_E}{N} = P_E \qquad \text{无偏的}$$

$$\sigma_{\hat{P}_e}^2 = \frac{\sigma_{N_e}^2}{N^2} = \frac{P_E(1-P_E)}{N} \qquad \text{一致的}$$

3.1 二项式分布

讨论：

- 估计器是无偏的---蒙特卡罗仿真可以在 平均意义下提供正确的结果
- 估计器是一致的---估计值会以较高的概率落在参数真值的附近

- 为达到给定的方差，必须记录到一定数量的差错，这反过来又要求一定的仿真时间。

- 存在的问题：无法事先确定所需要的试验次数N。
- 解决办法：通过运用界或者其他分析方法，将P_E的值确定在一个数量级之内。

3.1 二项式分布

例：在差错事件相互独立的情况下，可以用投掷硬币事件来模拟二进制发送.（代码见c202.m）

- N个符号的发送可建模为对一枚不均匀硬币进行N次投掷。可以假设第i次投掷结果为“反面”相当于对第i次发送作出正确的判断，而第i次投掷结果为“正面”相当于对此次发送作出错误的判决。

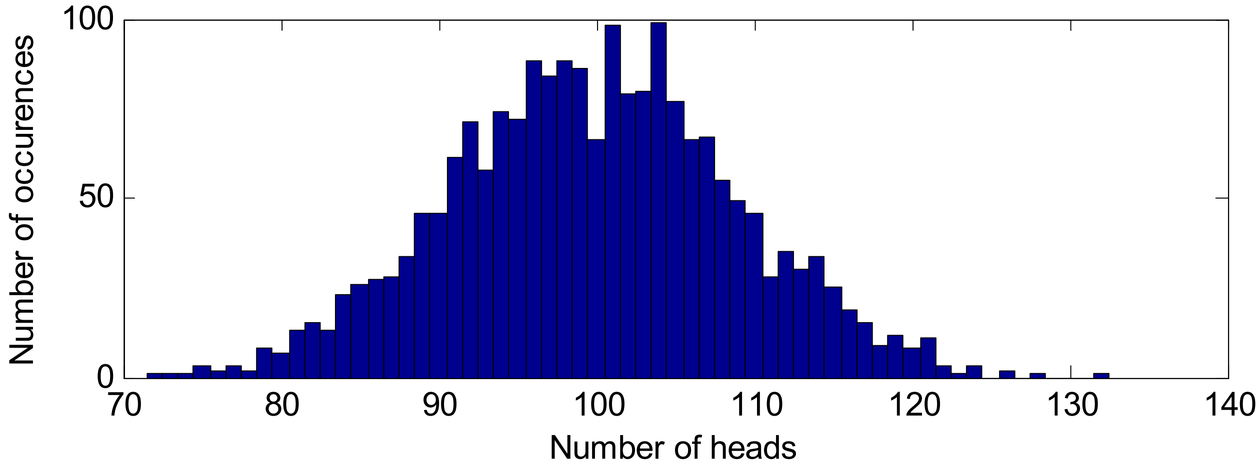
- 因为投掷是相互独立的，所以这个实验模拟了AWGN信道中的二进制数据传输。

3.1 二项式分布

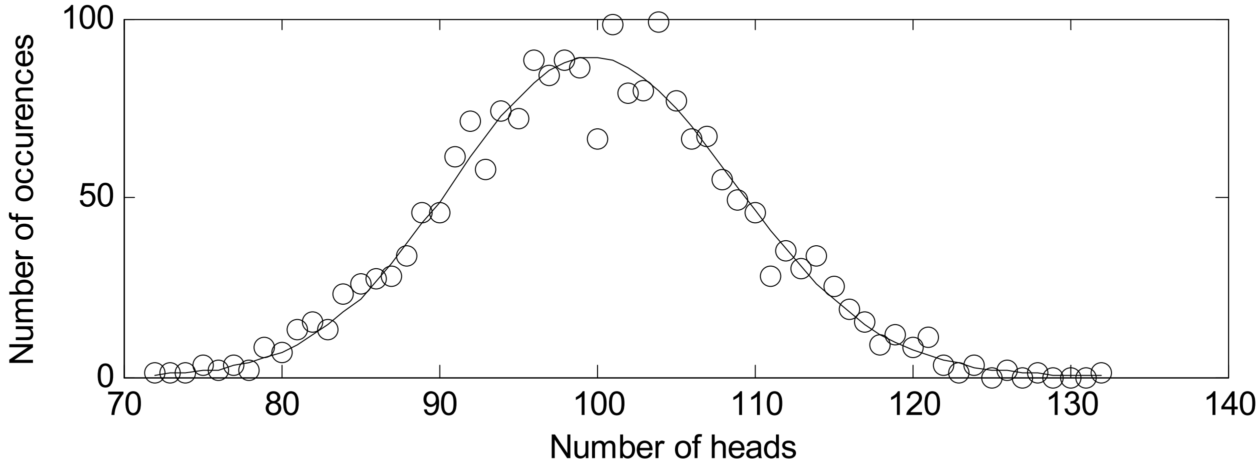
```
M = 2000;              % number of experiments
N = 500;               % Number of tosses / experiment
H = zeros(1,M);        % Initialize array
H_theor = zeros(1,M);  % Initialize array
for j=1:M
    A = rand(1,N);
    heads = 0;
    for k=1:N
        if A(k)<=0.2
            heads = heads+1;
        end
    end
    H(j) = heads;
end
H_max = max(H); H_min = min(H);
r = H_min:H_max;
[Nb] = hist(H,r);
%
for k=H_min:H_max
    H_theor(k) = M*nkchoose(N,k)*((0.2)^k)*((0.8)^(N-k));
end
```

```
r = H_min:H_max;
[Nb] = hist(H,r);
%
for k=H_min:H_max
    H_theor(k) = M*nkchoose(N,k)*((0.2)^k)*((0.8)^(N-k));
end
subplot(2,1,1)
hist(H,r)
xlabel('Number of heads')
ylabel('Number of occurences')
subplot(2,1,2)
plot(r,Nb,'ok',r,H_theor(1,H_min:H_max),'k')
xlabel('Number of heads')
ylabel('Number of occurences')
% End of script file.
```

3.1 二项式分布



100
50
0
Number of occurences
70
80
90
100
110
120
130
140
Number of heads



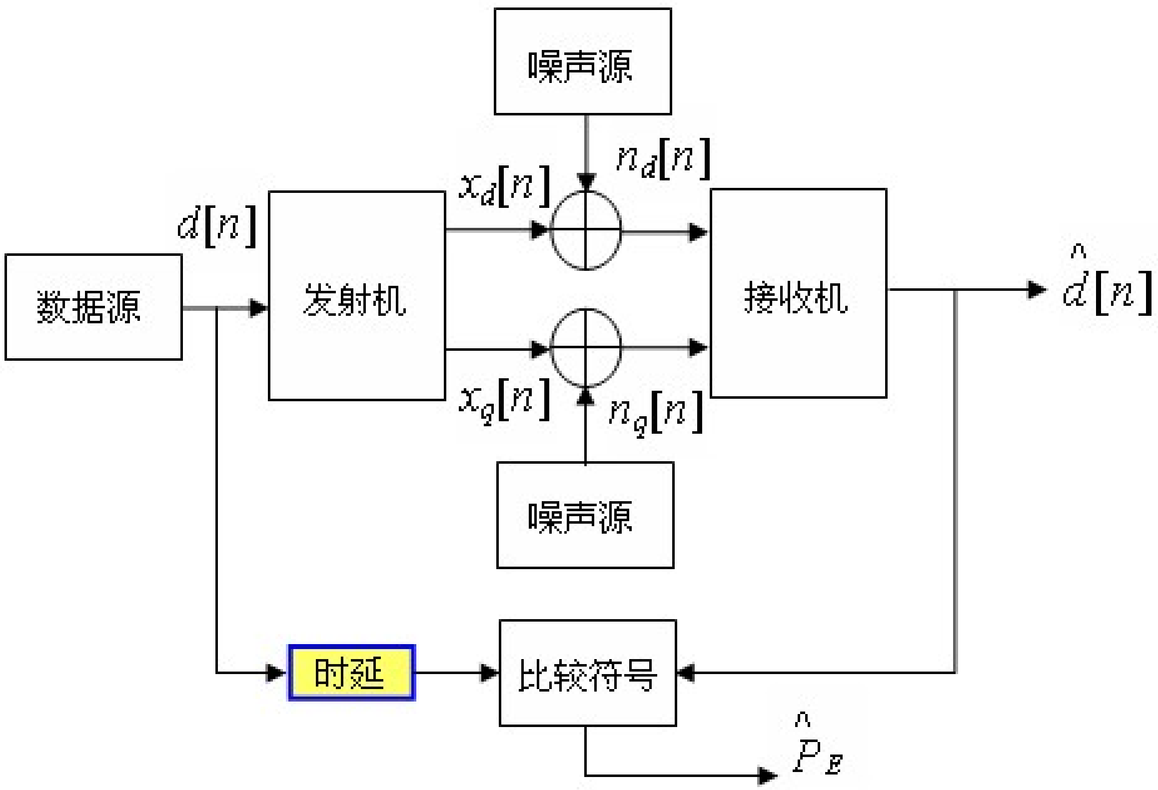
100
50
0
Number of occurences
70
80
90
100
110
120
130
140
Number of heads

3.2 两个简单的蒙特卡罗仿真

通信系统的蒙特卡罗仿真，先作以下假设：

- 在发射机中没有进行脉冲成形
- 假设信道是AWGN信道
- 信源输出端的数据符号是相互独立和等概率的
- 在系统中没有滤波处理，因此不存在码间干扰
- 时延可以不考虑

3.2 两个简单的蒙特卡罗仿真



简单通信系统的仿真模型

由于没有滤波，系统的时延为零，因此，在比较发送符号和接收符号之前，用于将相应符号进行对齐或同步的时延模块，对于这里的仿真是没有必要的。但是几乎在所有的仿真中都需要这个重要的部分

3.1 两个简单的蒙特卡罗仿真

调制器输出端的带通信号

$$x(t,n) = A_c cos[2\pi f_c t + k_m d[n] + \theta]$$

其中Ac表示载波副值，Km是跟调制有关的常数，的d[n]是第n个数据符号(d[n] =0或1)，可知x(t,n)的复包络只是符号序数n的函数，表示为

$$\tilde{x}[n] = A_c exp[k_m d[n] + \theta]$$

- 同相和正交分量（θ=0）

$$x_d[n] = A_c cos(k_m d[n]) \qquad x_q[n] = A_c sin(k_m d[n])$$

3.1 两个简单的蒙特卡罗仿真

- 确定以E_b/N_0为函数的BER

  保持E_b恒定，让噪声功率在感兴趣的范围内递增，噪声方差和噪声功率谱密度PSD的关系是

$$\sigma_n^2 = \frac{N_0 f_s}{2} \qquad N_0 = \sigma_n^2 \frac{2}{f_s}$$

- f_s为采样频率，且能量E_b和采样频率都归一化为1。

$$SNR = E_b / N_0 = \frac{f_s}{2}\frac{E_b}{\sigma_n^2} \qquad \sigma_n = \sqrt{\frac{1}{2SNR}}$$

3.2 两个简单的蒙特卡罗仿真

- 确定任意的功率谱密度或自相关函数

要确定一个随机数序列，使之具有给定自相关函数或等价地具有给定的功率谱密度，一个通用的方法是，对一组不相关的样本进行适当的滤波，从而使之具有目标功率谱密度。根据定义，不相关样本的功率谱密度在仿真带宽<fs/2上是常数，而方差就是Sn(f)曲线下的面积

$$\sigma_n^2 = \frac{N_0 f_s}{2} \qquad N_0 = \sigma_n^2 \frac{2}{f_s}$$

因此，为了产生功率谱密度为No的噪声，随机数(噪声)发生器的方差和采样频率必须满足 $f_s = 2\sigma_n^2/N_0$

可见，给定功率谱密度所要求的噪声发生器的方差是采样频率的函数。

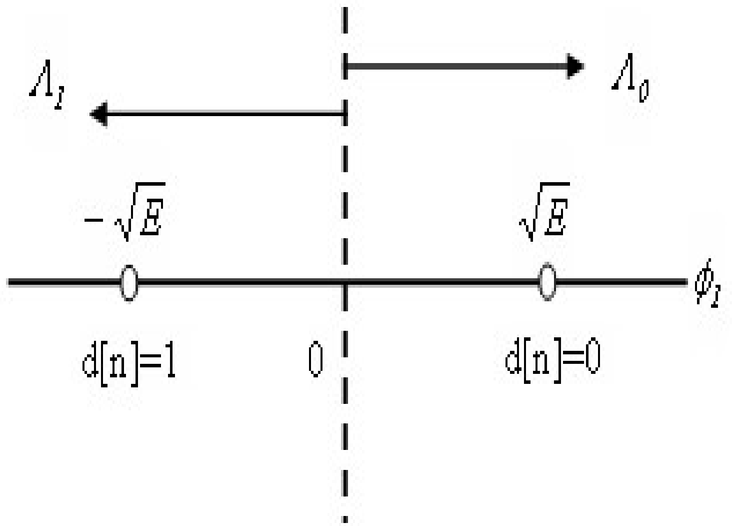
3.2 两个简单的蒙特卡罗仿真

- 例1：BPSK 令A_c=1和k_m= π， 由此可得（代码见c203.m ）

$$x_d[n] = \cos(\pi d[n]) = \begin{cases} 1, & d[n]=0 \\ -1, & d[n]=1 \end{cases}$$

$$x_q[n] = \sin(\pi d[n]) = 0$$

BPSK信号的空间表示



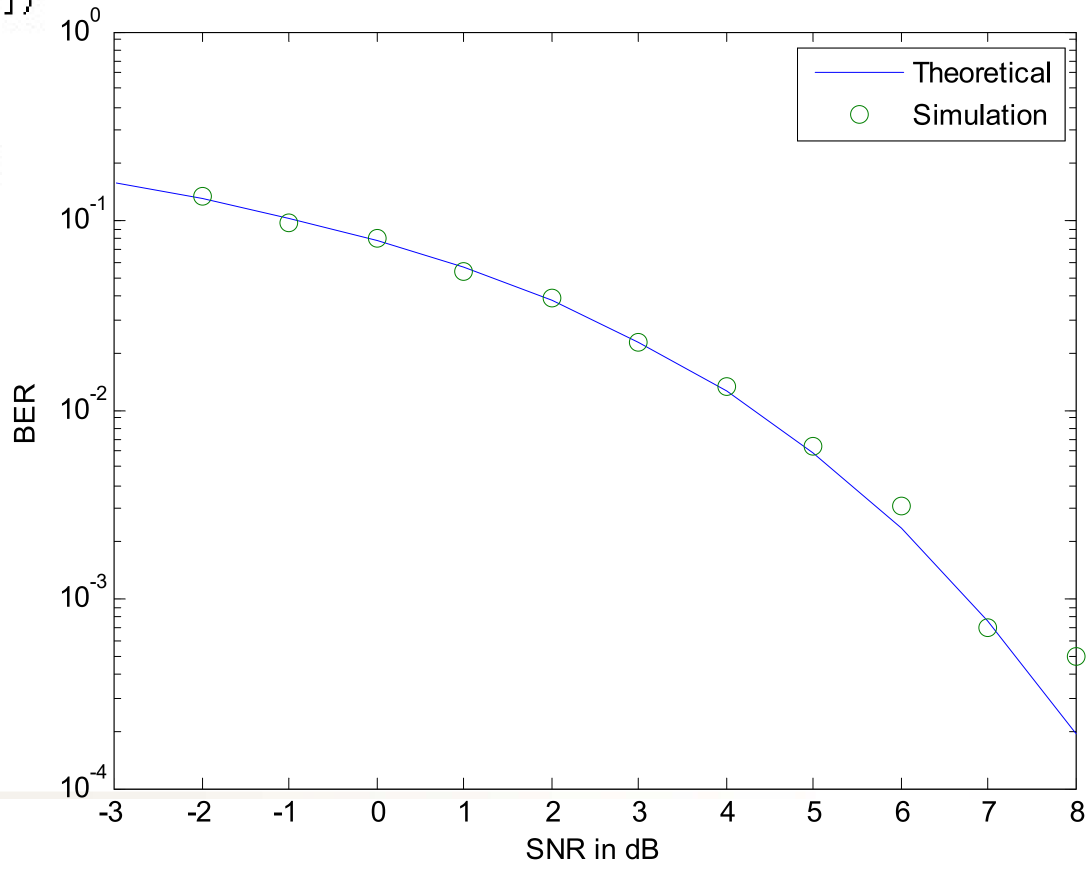
判决规则

$$\hat{d}[n] = \begin{cases} 0, & y_d[n] > 0 \\ 1, & y_d[n] < 0 \end{cases}$$

```
    snrdB_min = -3; snrdB_max = 8;        % SNR (in dB) limits
    snrdB = snrdB_min:1:snrdB_max;
    Nsymbols = input('Enter number of symbols > ');
    snr = 10.^(snrdB/10);                  % convert from dB
    h = waitbar(0,'SNR Iteration');
    len_snr = length(snrdB);
for j=1:len_snr                         % increment SNR
    waitbar(j/len_snr)
    sigma = sqrt(1/(2*snr(j)));          % noise standard deviation
    error_count = 0;
    for k=1:Nsymbols                     % simulation loop begins
        d = round(rand(1));              % data
        x_d = 2*d - 1;                   % transmitter output
        n_d = sigma*randn(1);            % noise
        y_d = x_d + n_d;                 % receiver input
        if y_d > 0                       % test condition
            d_est = 1;                   % conditional data estimate
        else
            d_est = 0;                   % conditional data estimate
        end
        if (d_est ~= d)
            error_count = error_count + 1; % error counter
        end
    end                                      % simulation loop ends
    errors(j) = error_count;                 % store error count for plot
end
```

3.2 两个简单的蒙特卡罗仿真

```
close(h)
ber_sim = errors/Nsymbols;                  % BER estimate
ber_theor = qfunc(sqrt(2*snr));                 % theoretical BER
semilogy(snrdB,ber_theor,snrdB,ber_sim,'o')
axis([snrdB_min snrdB_max 0.0001 1])
xlabel('SNR in dB')
ylabel('BER')
legend('Theoretical','Simulation')
```

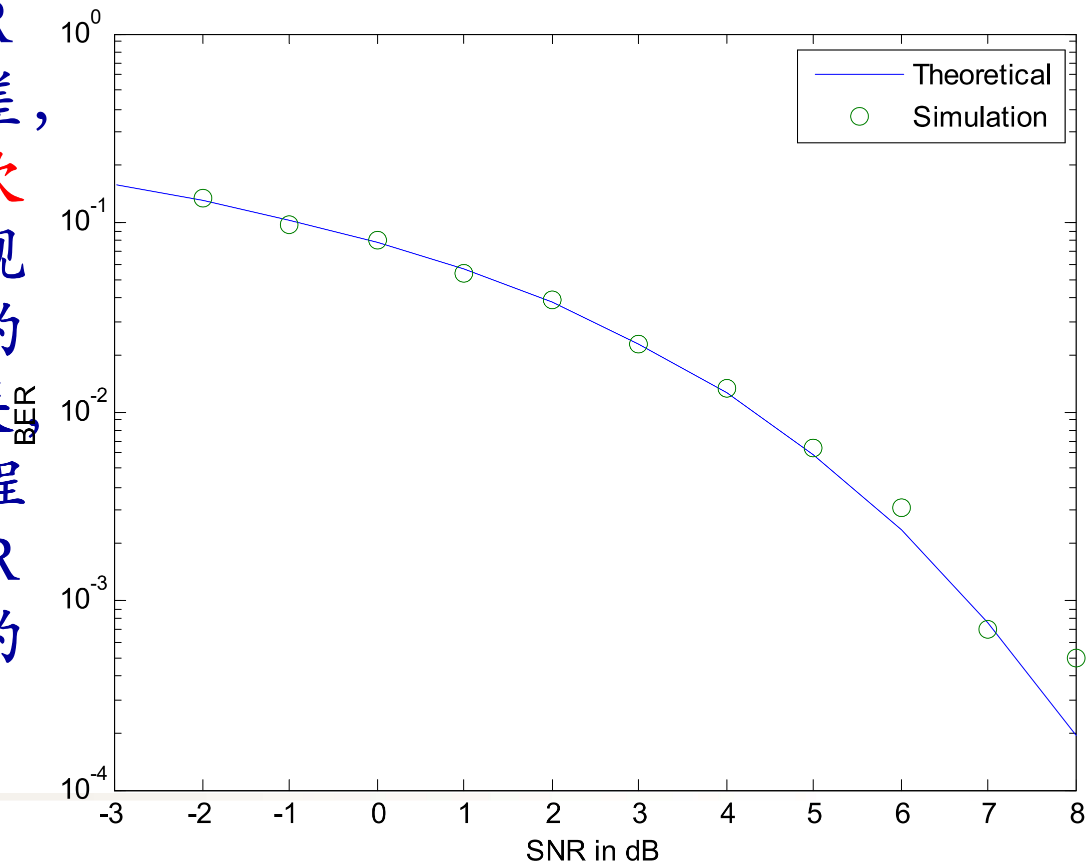


3.2 两个简单的蒙特卡罗仿真

小实验：改变C203.m中的输入点数，对结果会有什么影响?

3.2 两个简单的蒙特卡罗仿真

对于每一个SNR值，发送Nsymbols个符号。注意当SNR增加时，BER估计器的可靠度会变差，这是由于差错发生的次数减少的缘故。这一现象表明，仿真所发送的符号数可能与SNR有关，或者是连续运行仿真程序，直到对每一个SNR值都记录到相同数目的差错为止。



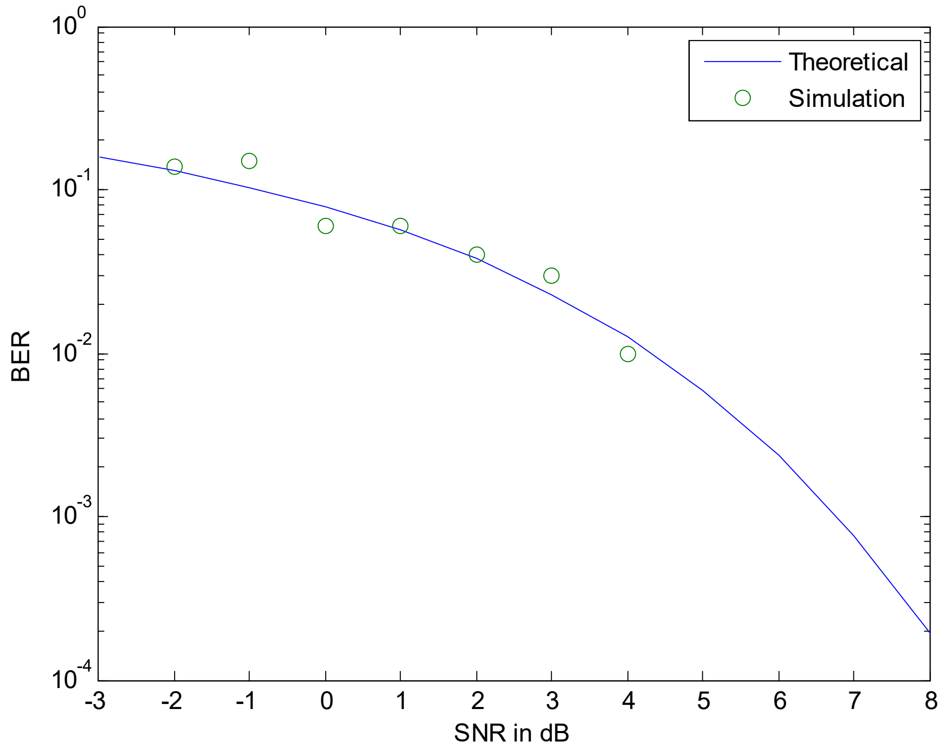
3.2 两个简单的蒙特卡罗仿真

```
Enter number of symbols > 10000
>> c203_MCBPSK
Enter number of symbols > 100
>>
```

若Pe=10^{-3}，那么N远远大于1000次。当N=10000时，我们认为这个样本数是得到Pe的可靠估计的最小值。所以作为经验公式，当

Pe《1时，样本数应满足条件：

$$N > \frac{10}{P_e}$$



3.2 两个简单的蒙特卡罗仿真

waitbar

许多蒙特卡洛仿真需要数小时，甚至数天才能完成，一个好做法是为仿真者提供必要的信息，以便让他们确信仿真还爱运行，如果可能的话，提供一些用于洞悉仿真所需要运行时间的信息也是很有用处的

3.2 两个简单的蒙特卡罗仿真

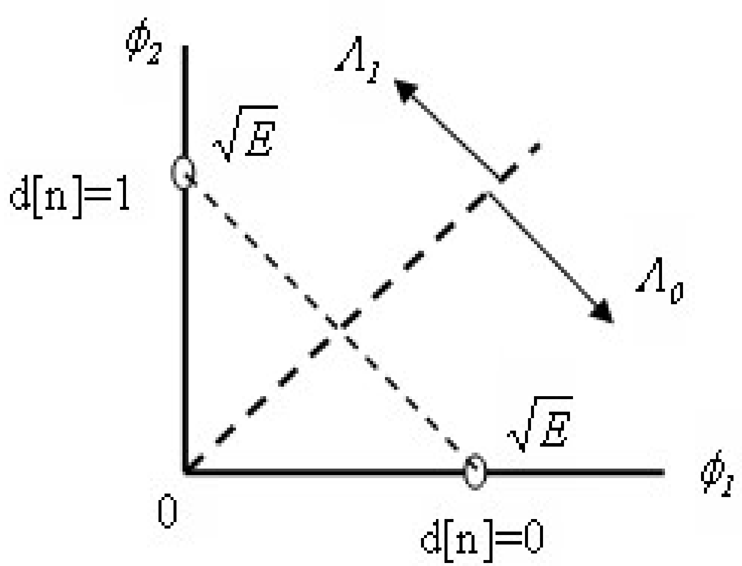
```
h=waitbar(0,'please wait...');
set(h,'doublebuffer','on');
for i=1:100,
if i>=88
waitbar(i/100,h,'计算即将完成')
pause(0.05);
else
str=['正在计算中',num2str(i),'%...'];
waitbar(i/100,h,str);
pause(0.05);
end
end
close(h);
```

3.2 两个简单的蒙特卡罗仿真

- 例2：BFSK Ac=1和km=π/2，由此可得（代码见c204.m）

$$x_d[n] = cos(\frac{\pi}{2}d[n]) = \begin{cases} 1, & d[n]=0 \\ 0, & d[n]=1 \end{cases}$$

$$x_q[n] = sin(\frac{\pi}{2}d[n]) = \begin{cases} 0, & d[n]=0 \\ 1, & d[n]=1 \end{cases}$$



判决规则

$$\hat{d}[n] = \begin{cases} 0, & y_d[n] > y_q[n] \\ 1, & y_d[n] < y_q[n] \end{cases}$$

```
clear all
snrdB_min = 0; snrdB_max = 10;          % SNR (in dB)limits
snrdB = snrdB_min:1:snrdB_max;
Nsymbols = input('Enter number of symbols > ');
snr = 10.^(snrdB/10);                   % convert from dB
h = waitbar(0,'SNR Iteration');
len_snr = length(snrdB);
for j=1:len_snr                         % increment SNR
   waitbar(j/len_snr)
   sigma = sqrt(1/(2*snr(j)));          % noise standard deviation
   error_count = 0;
   for k=1:Nsymbols                     % simulation loop begins
      d = round(rand(1));               % data
      if d ==0
         x_d = 1;                       % direct transmitter output
         x_q = 0;                       % quadrature transmitter output
      else
         x_d = 0;                       % direct transmitter output
         x_q = 1;                       % quadrature transmitter output
      end
      n_d = sigma*randn(1);             % direct noise component
      n_q = sigma*randn(1);             % quadrature noise component
      y_d = x_d + n_d;                  % direct receiver input
      y_q = x_q + n_q;                  % quadrature receiver input
```

```
        if y_d > y_q                         % test condition
            d_est = 0;                       % conditional data estimate
        else
            d_est = 1;                       % conditional data estimate
        end
        if (d_est ~= d)
            error_count = error_count + 1;   % error counter
        end
    end                                      % simulation loop ends
    errors(j) = error_count;                 % store error count for plot
end
close(h)
ber_sim = errors/Nsymbols;                   % BER estimate
ber_theor = qfunc(sqrt(snr));                    % theoretical BER
semilogy(snrdB,ber_theor,snrdB,ber_sim,'o')
axis([snrdB_min snrdB_max 0.0001 1])
xlabel('SNR in dB')
ylabel('BER')
legend('Theoretical','Simulation')
% End of script file.
```

3.2 两个简单的蒙特卡罗仿真