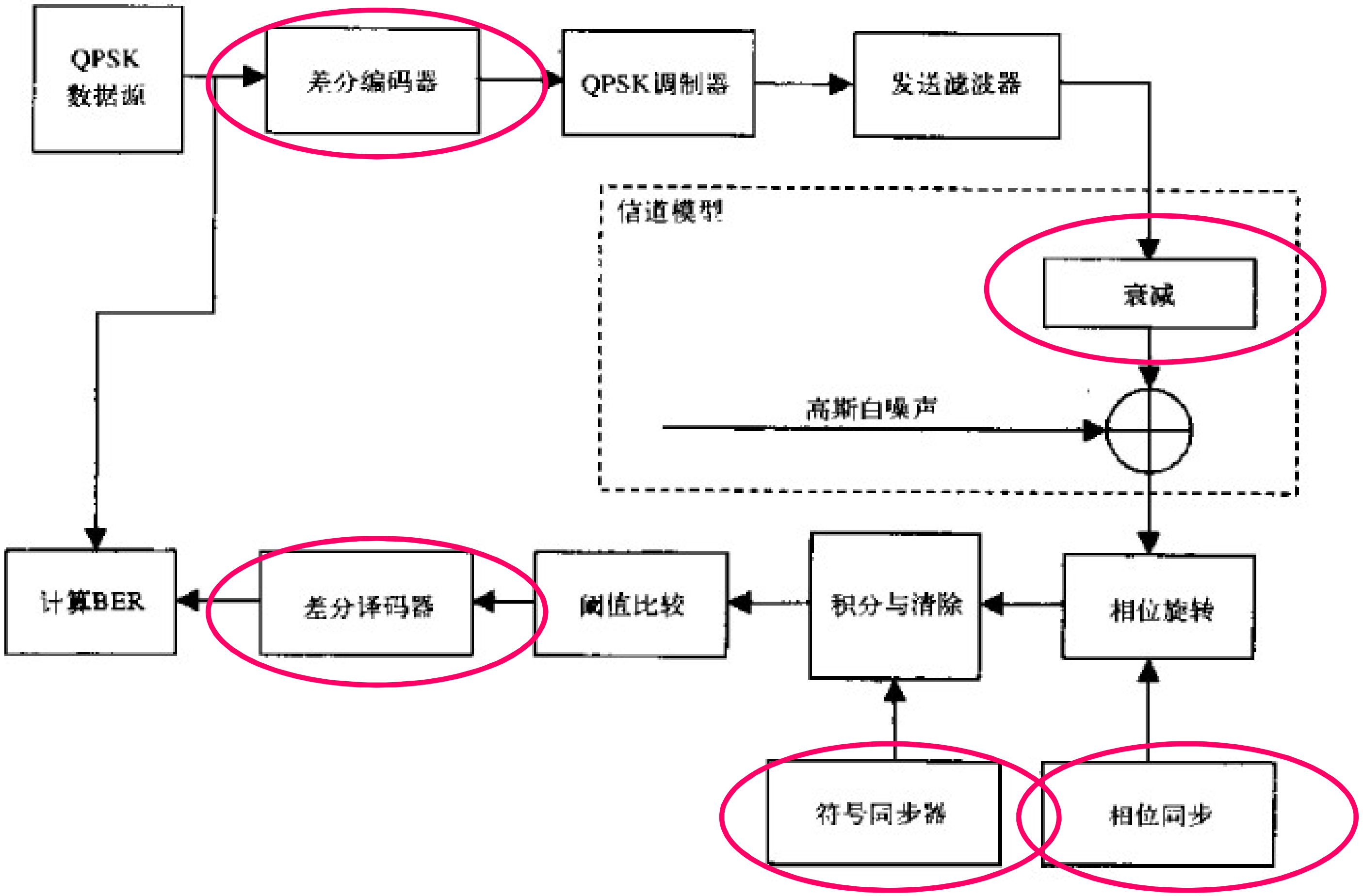
# 通信系统仿真
# 第2章 蒙特卡罗方法03

何晨光
哈尔滨工业大学
电子与信息工程学院

# 4 蒙特卡罗积分

例7：QPSK蒙特卡罗仿真



# 4 蒙特卡罗积分

例7：QPSK蒙特卡罗仿真

对于相干射频系统，接收机必须具备提供载波和符号同步的功能，但噪声和信道失真使接收机不可能完美地实现载波和符号同步。

不正确的载波同步将会导致发送信号相对于接收信号产生相位误差或相位旋转；
不正确的符号同步会导致积分-清楚检测器在不正确的时间区间上处理接收信号。

# 4 蒙特卡罗积分

例7：QPSK蒙特卡罗仿真仿真

BPSK和QPSK系统在解调时都存在相位模糊的问题。由于信道造成位置的信号时延，所以接收机不可能确定发送信号的绝对相位，从而发生相位模糊问题。

传统的解决这个问题的办法是不把信息位的编码包含在绝对相位中，而是包含在符号间的相位差中，实现差分编码DQPSK。

# 例7：QPSK – 确定系统时延

```
Eb = 23; No = -50;                % Eb (dBm) and No (dBm/Hz)
ChannelAttenuation = 70;          % channel attenuation in dB
N = 1000;
delay = -0.1:0.1:0.5;
EbNo = 10.^(((Eb-ChannelAttenuation)-No)/10);
BER_MC = zeros(size(delay));
for k=1:length(delay)
  BER_MC(k) = c214_MCQPSKrun(N,Eb,...
       No,ChannelAttenuation,delay(k),0,0,0);
  disp(['Simulation ',...
       num2str(k*100/length(delay)),'% Complete']);
end
BER_I = 0.5*erfc(sqrt(EbNo))*ones(size(delay)); % Theoretical BER
semilogy(delay,BER_MC,'o',delay,2*BER_I,'-')     %Plot BER vs Delay
xlabel('Delay (symbols)'); ylabel('Bit Error Rate');
legend('MC BER Estimate','Theoretical BER');grid;
% End of script file.
```

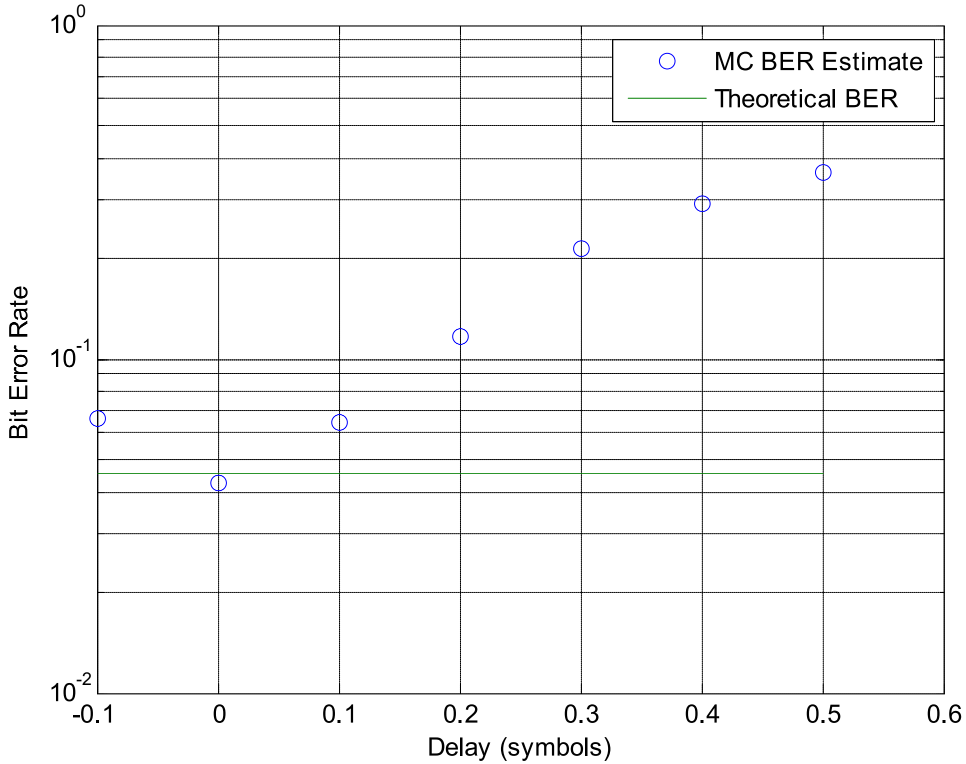
（代码见c215.m）

# 例7：QPSK – 确定系统时延

```
for k=1:length(delay)
    BER_MC(k) = c214_MCQPSKrun(N,Eb,...
            No,ChannelAttenuation,delay(k),0,0,0);
    disp(['Simulation ',...
            num2str(k*100/length(delay)),'% Complete']);
end
```

```
>> c215_MCQPSKdelay
Simulation 14.2857% Complete
Simulation 28.5714% Complete
Simulation 42.8571% Complete
Simulation 57.1429% Complete
Simulation 71.4286% Complete
Simulation 85.7143% Complete
Simulation 100% Complete
```

# 例7：QPSK – 确定系统时延



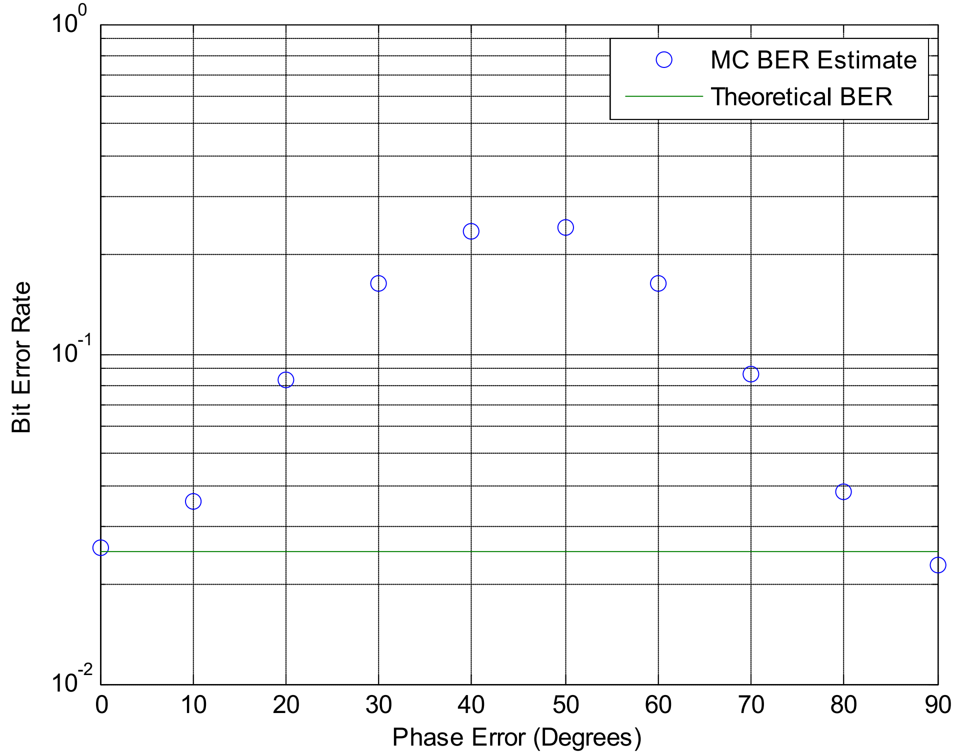

因为没有使用信道滤波器，最优时延是零符号周期。此仿真用的是符号周期来测量时延，而不是采样周期

# 例7：QPSK – 同步相位误差

测量误比特率对静态同步相位误差的灵敏度。当相位误差以10度为增量从0度变化到90度，完成灵敏度的测算。（代码见c215.m）

```
PhaseError = 0:10:90;                    % Phase Error at Receiver
Eb = 24; No = -50;                       % Eb (dBm) and No (dBm/Hz)
ChannelAttenuation = 70;                 % dB
EbNo = 10.^((Eb-ChannelAttenuation-No)/10);
BER_T = 0.5*erfc(sqrt(EbNo)*ones(size(PhaseError)));
N = round(100./BER_T);
BER_MC = zeros(size(PhaseError));
for k=1:length(PhaseError)
  BER_MC(k) = c214_MCQPSKrun(N(k),Eb,No,ChannelAttenuation,0,0,...
    PhaseError(k),0);
  disp(['Simulation ',num2str(k*100/length(PhaseError)),'% Complete']);
end
semilogy(PhaseError,BER_MC,'o',PhaseError,2*BER_T,'-')
xlabel('Phase Error (Degrees)');
ylabel('Bit Error Rate');
legend('MC BER Estimate','Theoretical BER'); grid;
```

# 例7：QPSK – 同步相位误差




如图所示，仿真中所得到的误比特率在相位误差为45度的时候达到最大，然后在相位误差为0度或者90度时下降到最优值（零同步相位误差时的值）这也是差分编码所带来的结果。

# 例7：QPSK – BER子函数

```
function BER_MC=MCQPSKrun(N,Eb,No,ChanAtt,...
    TimingBias,TimingJitter,PhaseBias,PhaseJitter)
fs = 1e+6;                                   % sampling Rate (samples/second)
SymRate = 1e+5;                              % symbol rate (symbols/second)
Ts = 1/fs;                                   % sampling period
TSym = 1/SymRate;                            % symbol period
SymToSend = N;                               % symbols to be transmitted
ChanBW = 4.99e+5;                            % bandwidth of channel (Hz)
MeanCarrierPhaseError = PhaseBias;           % mean of carrier phase
StdCarrierPhaseError = PhaseJitter;          % stdev of phese error
MeanSymbolSyncError = TimingBias;            % mean of symbol sync error
StdSymbolSyncError = TimingJitter;           % stdev of symbol sync error
ChanGain = 10^(-ChanAtt/20);                 % channel gain (linear units)
TxBitClock = Ts/2;                           % transmitter bit clock
RxBitClock = Ts/2;                           % reciever bit clock
```

（代码见c214.m）

# 例7：QPSK – BER子函数

```
function BER_MC=MCQPSKrun(N,Eb,No,ChanAtt,...
    TimingBias,TimingJitter,PhaseBias,PhaseJitter)
```

QPSK的BER子函数包含8个输入和1个输出

输出：

BER_MC 比特率

输入：

N 发送符号个数

$E_b$ 比特能量（dB）

$N_0$ 噪声功率（dB）

ChanAtt 信道衰减（dB）

TimingBias 符号同步误差均值

TimingJitter 符号同步抖动值

PhaseBias 相位同步误差均值

PhaseJitter 相位同步误差均值

（代码见c214.m）

# 例7：QPSK – BER子函数

```
function BER_MC=MCQPSKrun(N,Eb,No,ChanAtt,...
    TimingBias,TimingJitter,PhaseBias,PhaseJitter)
fs = 1e+6;                          % sampling Rate (samples/second)
SymRate = 1e+5;                     % symbol rate (symbols/second)
Ts = 1/fs;                          % sampling period
TSym = 1/SymRate;                   % symbol period
SymToSend = N;                      % symbols to be transmitted
ChanBW = 4.99e+5;                   % bandwidth of channel (Hz)
```

fs 采样速率　　SymRate　符号速率

两者的比值：

SampPerSym = fs/SymRate;

是每符号采样的个数

Ts　　采样周期

Tsym　　符号周期

ChanBW　带宽

（代码见c214.m）

# 例7：QPSK – BER子函数

```
MeanCarrierPhaseError = PhaseBias;        % mean of carrier phase
StdCarrierPhaseError = PhaseJitter;       % stdev of phese error
MeanSymbolSyncError = TimingBias;         % mean of symbol sync error
StdSymbolSyncError = TimingJitter;        % stdev of symbol sync error
```

符号载波同步和相位同步的均值和标准差，也就是基准值和抖动情况

```
TxBitClock = Ts/2;                        % transmitter bit clock
RxBitClock = Ts/2;                        % reciever bit clock
```

发射机比特时钟
接收机比特时钟
用来发射和接收比特数据

（代码见c214.m）

# 例7：QPSK – BER子函数

```
ChanGain = 10^(-ChanAtt/20);                  % channel gain (linear units)
RxNoiseStd = sqrt((10^((No-30)/10))*(fs/2));          % stdev of noise
TxSigAmp = sqrt(10^((Eb-30)/10)*SymRate);             % signal amplitude
```

信道增益
噪声的标准差
发射信号的副值

（代码见c214.m）

# 例7：QPSK – BER子函数

```
% Allocate some memory for probes.
%
SampPerSym = fs/SymRate;
probe1 = zeros((SymToSend+1)*SampPerSym,1);
probe1counter = 1;
probe2 = zeros((SymToSend+1)*SampPerSym,1);
probe2counter = 1;
```

每符号采样的个数
两个用于接收的缓存区，长度为抽样后的长度
probe1  probe2counter
数组和相应的索引

（代码见c214.m）

# 例7：QPSK – BER子函数

```
% Counters to keep track of how many symbols have have been sent.
%
TxSymSent = 1;
RxSymDemod = 0;
%
% Buffers that contain the transmitted and received data.
%
[unused,SourceBitsI] = random_binary(SymToSend,1);
[unused,SourceBitsQ] = random_binary(SymToSend,1);
```

发送和接收符号的计数器

I,Q两路分别产生二进制随机数0和1

（代码见c214.m）

# 例7：QPSK – BER子函数

```
% Differentially encode the transmitted data.
%
IxBitsI = SourceBitsI*0;
IxBitsQ = SourceBitsQ*0;
for k=2:length(IxBitsI)
    IxBitsI(k) = or(and(not(xor(SourceBitsI(k),SourceBitsQ(k))),...
                 xor(SourceBitsI(k),IxBitsI(k-1))), ...
                 and(xor(SourceBitsI(k),SourceBitsQ(k)),...
                 xor(SourceBitsQ(k),IxBitsQ(k-1))));
    IxBitsQ(k) = or(and(not(xor(SourceBitsI(k),SourceBitsQ(k))),...
                 xor(SourceBitsQ(k),IxBitsQ(k-1))), ...
                 and(xor(SourceBitsI(k),SourceBitsQ(k)),...
                 xor(SourceBitsI(k),IxBitsI(k-1))));
end
```

设置起始位为0

I,Q两路分别进行差分编码

（代码见c214.m）

# 例7：QPSK – BER子函数

```
% Differential decoder.
%
SinkBitsI = SourceBitsI*0;
SinkBitsQ = SourceBitsQ*0;
%
for k=2:RxSymDemod
    SinkBitsI(k) = or(and(not(xor(RxBitsI(k),RxBitsQ(k))),...
                   xor(RxBitsI(k),RxBitsI(k-1))),...
                   and(xor(RxBitsI(k),RxBitsQ(k)),...
                   xor(RxBitsQ(k),RxBitsQ(k-1))));
    SinkBitsQ(k) = or(and(not(xor(RxBitsI(k),RxBitsQ(k))),...
                   xor(RxBitsQ(k),RxBitsQ(k-1))),...
                   and(xor(RxBitsI(k),RxBitsQ(k)),...
                   xor(RxBitsI(k),RxBitsI(k-1))));
end
```

设置起始位为0

（代码见c214.m）

I,Q两路分别进行差分译码

# 例7：QPSK – BER子函数

```
% Make a complex data stream of the I and Q bits.
%
IxBits = ((IxBitsI*2)-1)+(sqrt(-1)*((IxBitsQ*2)-1));
%
RxIntegrator = 0;                    % initialize receiver integrator
IxBitClock = 2*ISym;                 % initialize transmitter
```

I,Q两路转换为+1和-1，然后形成复数

初始化接收端积分器
初始化发射机时钟

（代码见c214.m）

# 例7：QPSK – BER子函数

```
% Design the channel filter, and create the filter state array.
%
[b,a] = butter(2,ChanBW/(fs/2));
b=[1]; a=[1];                                   % filter bypassed
[junk,FilterState]=filter(b,a,0);
%
```

设置信道滤波器，这里的系数都是1
忽略滤波器的影响

（代码见c214.m）

# 例7：QPSK – BER子函数

```
while TxSymSent < SymToSend
    % Update the transmitter's clock, and see
    % if it is time to get new data bits
    TxBitClock=TxBitClock+Ts;
    if TxBitClock > TSym
        % Time to get new bits
        TxSymSent=TxSymSent+1;
        % We don't want the clock to increase off
        % to infinity, so subtract off an integer number
        % of Tb seconds
        TxBitClock=mod(TxBitClock,TSym);
        % Get the new bit, and scale it up appropriately.
        TxOutput=TxBits(TxSymSent)*TxSigAmp;
    end
```

大循环用的是while，执行完语句后发现条件仍然成立继续执行语句

Ts 是采样的周期，每个bit的来接收

确定接收的位置后，形成发送信号

（代码见c214.m）

# 例7：QPSK – BER子函数

```
% Pass the transmitted signal through the channel filter.
%
[Rx,FilterState]=filter(b,a,TxOutput,FilterState);
%
% Add white Gaussian noise to the signal.
%
Rx=(ChanGain*Rx)+(RxNoiseStd*(randn(1,1)+sqrt(-1)*randn(1,1)));
%
% Phase rotation due to receiver carrier synchronization error.
%
PhaseRotation = exp(sqrt(-1)*2*pi*...
    (MeanCarrierPhaseError+(randn(1,1)*StdCarrierPhaseError))/360);
Rx=Rx*PhaseRotation;
probe1(probe1counter)=Rx; probe1counter=probe1counter+1;
```

IQ两路都加噪声

增加相位同步误差的影响

（代码见c214.m）

# 例7：QPSK – BER子函数

```
% Update the Integrate and Dump Filter at the receiver.
%
RxIntegrator = RxIntegrator+Rx;
probe2(probe2counter) = RxIntegrator;
probe2counter = probe2counter+1;
```

采用缓存区进行积分的过程，形成统计量

（代码见c214.m）

# 例7：QPSK – BER子函数

```
% Update the receiver clock, to see if it is time to
% sample and dump the integrator.
%
RxBitClock = RxBitClock+Ts;
RxISym = ISym*(1+MeanSymbolSyncError+(StdSymbolSyncError*randn(1,1)));
if RxBitClock > RxISym                          % time to demodulate symbol
   RxSymDemod = RxSymDemod+1;
   RxBitsI(RxSymDemod) = round(sign(real(RxIntegrator))+1)/2;
   RxBitsQ(RxSymDemod) = round(sign(imag(RxIntegrator))+1)/2;
   RxBitClock = RxBitClock - ISym;          % reset receive clock
   RxIntegrator = 0;                        % reset integrator
end
end
```

采用时钟接进行接收

添加符号同步误差

从复数转换成IQ两路信号

（代码见c214.m）

# 例7：QPSK – BER子函数

```
% Look for best time delay between input and output for 100 bits.
%
[C,Lags] = vxcorr(SourceBitsI(10:110),SinkBitsI(10:110));
[MaxC,LocMaxC] = max(C);
BestLag = Lags(LocMaxC);
%
% Adjust time delay to match best lag
%
if BestLag > 0
    SourceBitsI = SourceBitsI(BestLag+1:length(SourceBitsI));
    SourceBitsQ = SourceBitsQ(BestLag+1:length(SourceBitsQ));
elseif BestLag < 0
    SinkBitsI = SinkBitsI(-BestLag+1:length(SinkBitsI));
    SinkBitsQ = SinkBitsQ(-BestLag+1:length(SinkBitsQ));
end
```

通过互相关计算时延

调整时延，符号对齐

（代码见c214.m）

# 例7：QPSK – BER子函数 互相关

对发送和信号进行互相关vxcorr，为了发送和接收的符号可以正确对准，以便更精确的确定误比特率

```
function [c,lags] = vxcorr(a,b)
a = a(:);                              % convert a to column vector
b = b(:);                              % convert b to column vector
M = length(a);                         % same as length(b)
maxlag = M-1;                          % maximum value of lag
lags = [-maxlag:maxlag]';              % vector of lags
A = fft(a,2^nextpow2(2*M-1));          % fft of A
B = fft(b,2^nextpow2(2*M-1));          % fft of B
c = ifft(A.*conj(B));                  % crosscorrelation
% Move negative lags before positive lags
c = [c(end-maxlag+1:end,1);c(1:maxlag+1,1)];
% Return row vector if a, b are row vectors
[nr nc]=size(a);
if(nr>nc)
    c=c.';
   lags=lags.';
end
```

# 例7：QPSK – BER子函数 互相关

```
a=[0,0,0,1,1,0];
b=[0,0,1,1,0,0];
[C,Lags] = vxcorr(a,b);
[aa,LocC] = max(C);
BestLag = Lags(LocC);
```

C

1x11 double

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3.4853e-17 | 8.3267e-17 | 1.1102e-16 | -1.1497e-17 | -1.0116e-16 | 1 | 2 | 1 | 4.3652e-17 | -8.3267e-17 | -1.1102e-16 |

Lags

1x11 double

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| -5 | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 | 5 |

aa

1x1 double

| | 1 |
|---|---|
| 1 | 2 |

LocC

1x1 double

| | 1 |
|---|---|
| 1 | 7 |

BestLag

1x1 double

| 1 |
|---|
| 1 |

# 例7：QPSK – BER子函数

调整长度进行对比

```
%
% Make all arrays the same length.
%
TotalBits = min(length(SourceBitsI),length(SinkBitsI));
TotalBits = TotalBits-20;
SourceBitsI = SourceBitsI(10:TotalBits);
SourceBitsQ = SourceBitsQ(10:TotalBits);
SinkBitsI = SinkBitsI(10:TotalBits);
SinkBitsQ = SinkBitsQ(10:TotalBits);
%
% Find the number of errors and the BER.
%
Errors = sum(SourceBitsI ~= SinkBitsI) + sum(SourceBitsQ ~= SinkBitsQ);
BER_MC = Errors/(2*length(SourceBitsI));
```

I,Q两路分别统计，然后累加

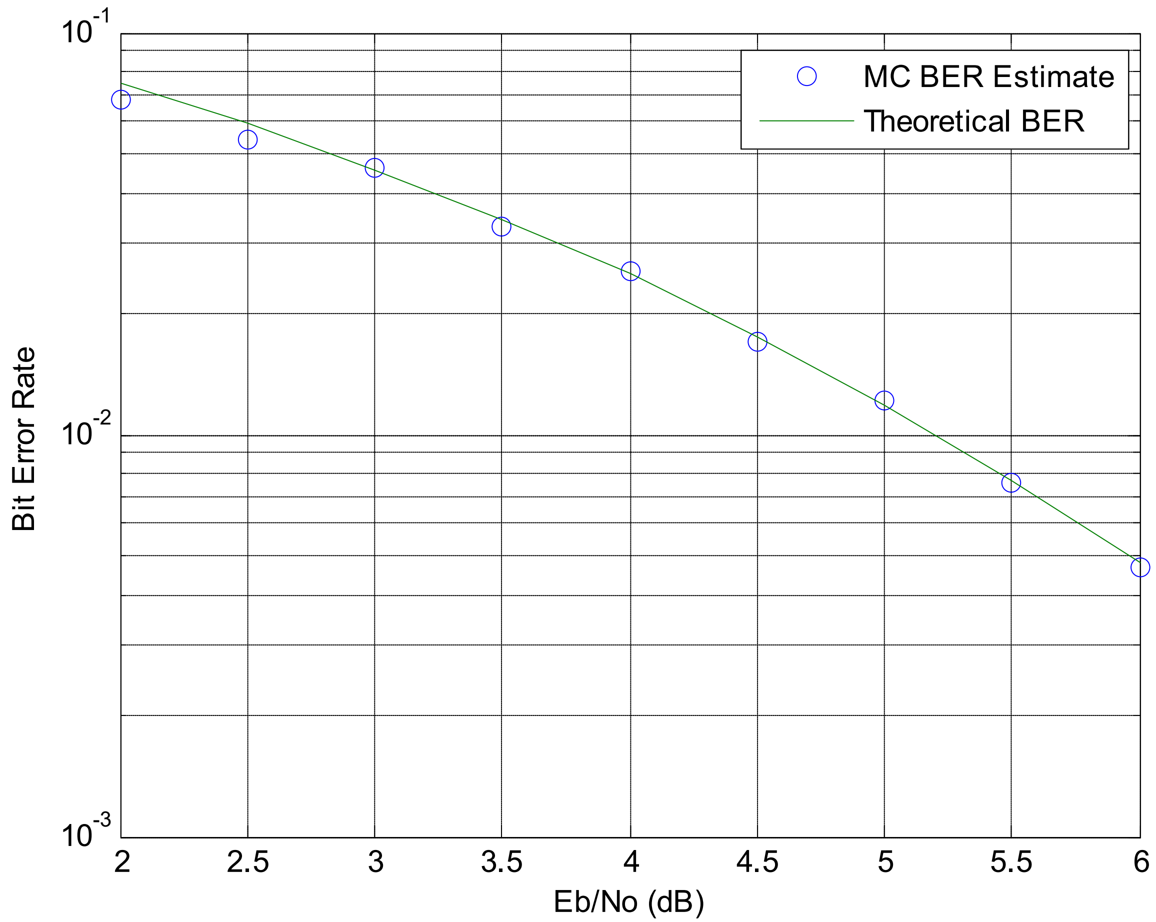
（代码见c214.m）

# 例7：主函数

```
Eb = 22:0.5:26; No = -50;                        % Eb (dBm) and No (dBm/Hz)
ChannelAttenuation = 70;                         % Channel attenuation in dB
EbNodB = (Eb-ChannelAttenuation)-No;             % Eb/No in dB
EbNo = 10.^(EbNodB./10);                         % Eb/No in linear units
BER_T = 0.5*erfc(sqrt(EbNo));                    % BER (theoretical)
N = round(100./BER_T);                           % Symbols to transmit
BER_MC = zeros(size(Eb));                        % Initialize BER vector
for k=1:length(Eb)                               % Main Loop
  BER_MC(k) = c214_MCQPSKrun(N(k),Eb(k),No,ChannelAttenuation,0,0,0,0);
  disp(['Simulation ',num2str(k*100/length(Eb)),'% Complete']);
end
semilogy(EbNodB,BER_MC,'o',EbNodB,2*BER_T,'-')
xlabel('Eb/No (dB)'); ylabel('Bit Error Rate');
legend('MC BER Estimate','Theoretical BER'); grid;
% End of script file.
```

（代码见c213.m）

# 例7：主函数




（代码见c213.m）

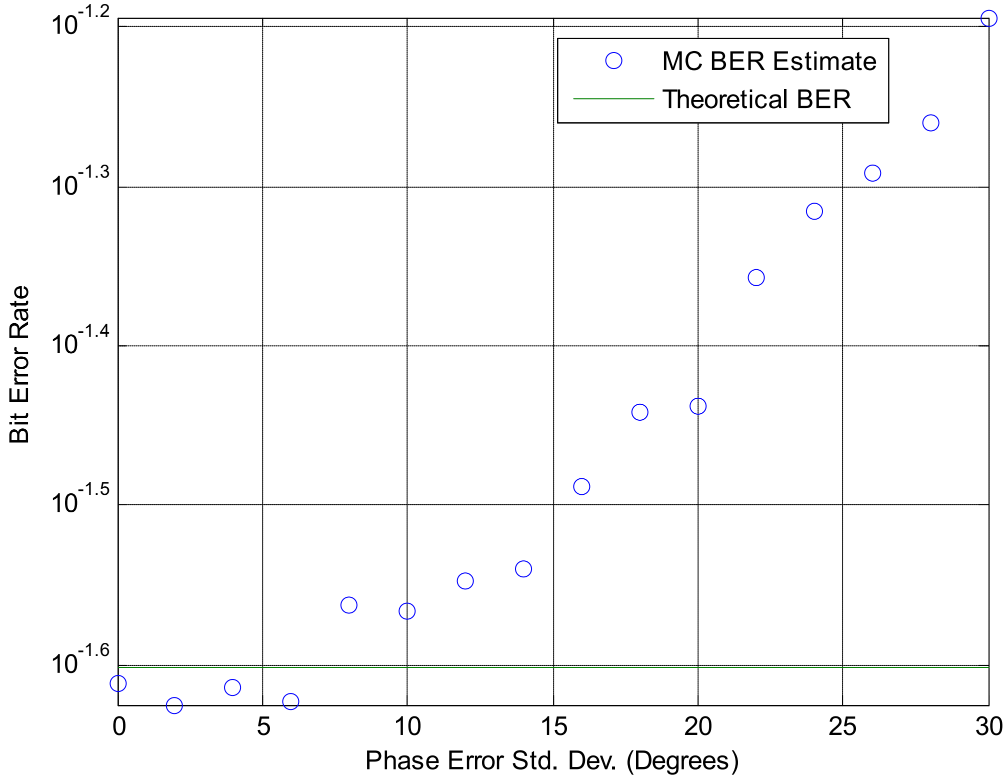
# 例7：相位抖动

考察相位抖动对BER的影响。
用白噪声来对相位误差过程进行建模

```
PhaseBias = 0; PhaseJitter = 0:2:30;
Eb = 24; No = -50;                          % Eb (dBm) and No (dBm/Hz)
ChannelAttenuation = 70;                    % dB
EbNo = 10.^((Eb-ChannelAttenuation-No)/10);
BER_T = 0.5*erfc(sqrt(EbNo)*ones(size(PhaseJitter)));
N = round(100./BER_T);
BER_MC = zeros(size(PhaseJitter));
for k=1:length(PhaseJitter)
  BER_MC(k) = c214_MCQPSKrun(N(k),Eb,No,ChannelAttenuation,0,0,...
    PhaseBias,PhaseJitter(k));
  disp(['Simulation ',num2str(k*100/length(PhaseJitter)),'% Complete']);
end
semilogy(PhaseJitter,BER_MC,'o',PhaseJitter,2*BER_T,'-')
xlabel('Phase Error Std. Dev. (Degrees)');
ylabel('Bit Error Rate');
legend('MC BER Estimate','Theoretical BER'); grid;
% End of script file.
```

（代码见c217.m）

# 例7：相位抖动




如图，当相位抖动的标准差增加时，误比特率也会增加。在许多系统的仿真中，用白噪声来建模相位抖动是不恰当的。可以设计一个有限冲激响应滤波器（FIR）来实现相位抖动过程中所需要的功率谱密度（PSD）

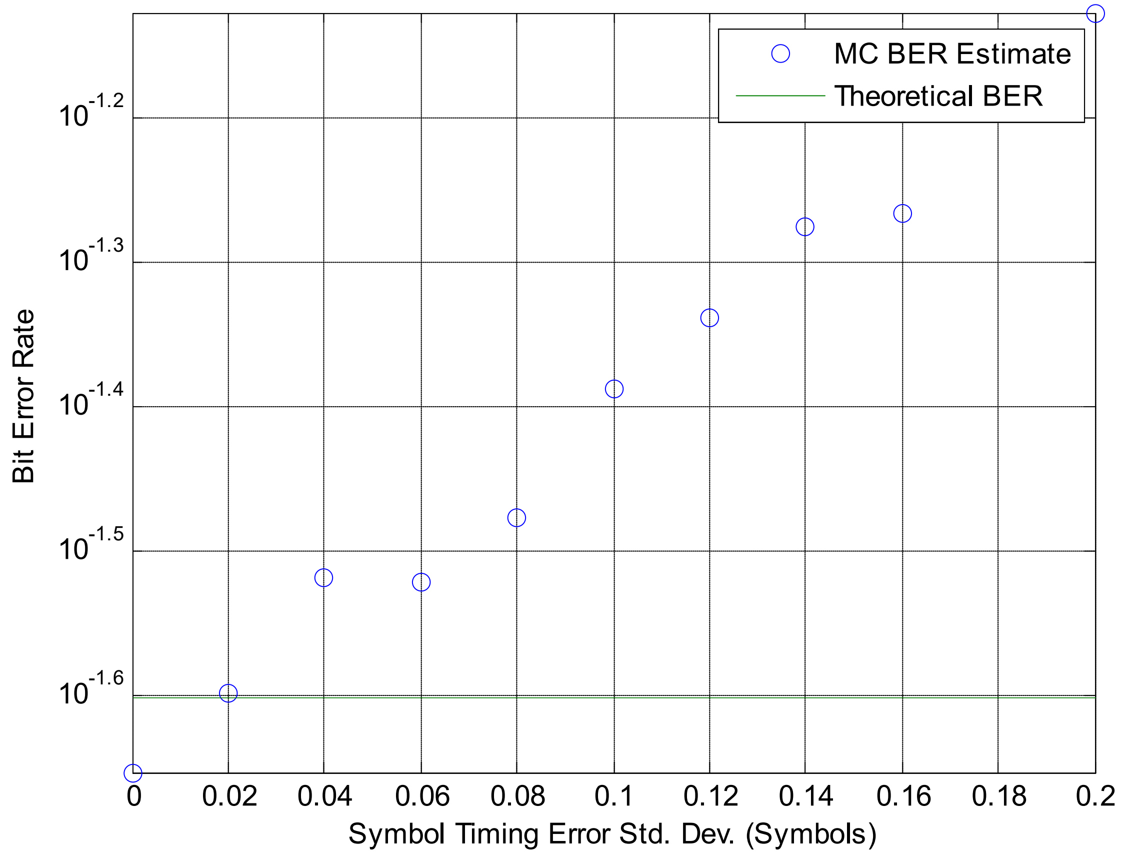
# 例7：符号同步抖动

考察符号同步抖动对BER的影响。
用白噪声来对符号同步误差过程进行建模

```
SymJitter = 0:0.02:0.2;
Eb = 24; No = -50;                  % Eb (dBm) and No (dBm/Hz)
ChannelAttenuation = 70;            % channel attenuation in dB
EbNo = 10.^((Eb-ChannelAttenuation-No)/10);
BER_I = 0.5*erfc(sqrt(EbNo)*ones(size(SymJitter)));
N=round(100./BER_I);
BER_MC = zeros(size(SymJitter));
for k=1:length(SymJitter)
  BER_MC(k) = c214_MCQPSKrun(N(k),Eb,No,ChannelAttenuation,0,SymJitter(k),0,0);
  disp(['Simulation ',num2str(k*100/length(SymJitter)),'% Complete']);
end
semilogy(SymJitter,BER_MC,'o',SymJitter,2*BER_I,'-')
xlabel('Symbol Timing Error Std. Dev. (Symbols)');
ylabel('Bit Error Rate');
legend('MC BER Estimate','Theoretical BER'); grid;
% End of script file.
```

（代码见c218.m）

# 例7：符号同步抖动




对符号抖动过程的记忆效应进行精确建模，也可以设计一个FIR滤波器实现所需要的功率谱密度PSD

习题5：例7考察了差分QPSK系统的蒙特卡罗仿真，重新编写QPSK系统而不是差分QPSK系统的仿真程序，与差分QPSK系统从多个方面（例如BER，符号同步误差，相位同步误差和抖动等方面）进行对比。