原 DVB-C系统中QAM调制与解调仿真

2015年02月21日 21:48:59　阅读数：10255

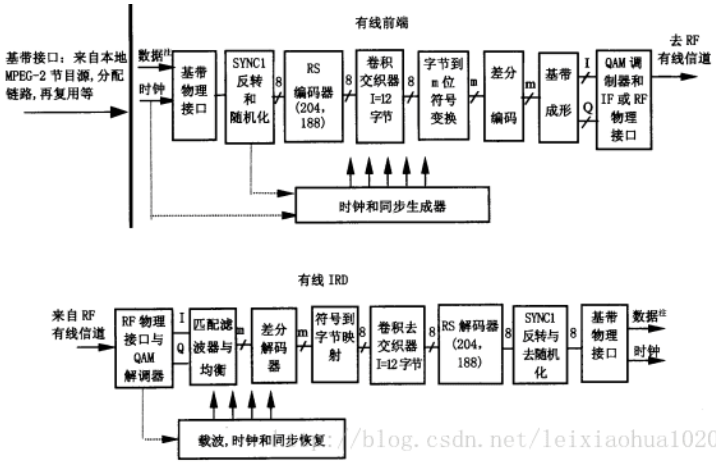
本文简单记录一下自己学习《通信原理》的时候调试的一个仿真DVB-C（Cable，数字有线电视）系统中QAM调制和解调的程序。自己一直是研究“信源”方面的东西，所以对“信道”这方面的知识进行实践的机会一直不是很多，做这个小程序的过程中也熟悉了不少相关的知识。在这个程序中，每执行一步操作，都会画出时域信号图和频域信号图，同时会在控制台打印出有关变量的取值，对于理解QAM调制与解调有一定的帮助。

一．DVB-C中QAM的调制与解调

简单介绍DVB-C系统中的QAM的调制与解调。DVB系列标准是数字电视领域最重要的标准，它是一个完整的数字广播解决方案，涉及数字电视广播的方方面面。DVB 规范了数字电视的系统结构和信号处理方式，各厂商可以开发各自的 DVB设备，只要该设备能够正确接收和处理信号并满足规范中所规定的性能指标就可以了。我国的卫星数字电视采用了DVB-S标准，地面广播数字电视采用了自主的DTMB标准，有线数字电视传输标准采用了DVB-C 标准。本文主要分析有线数字电视传输中的DVB-C标准。

1.DVB-C发送端和接收端

下面简单介绍DVB-C系统的发送端和接收端。DVB-C 标准描述了有线数字电视的帧结构，信道编码和调制。主要用于传送电视中的视频和音频信号。DVB-C 描述的有线数字电视前端与接收端的原理框图如图所示。

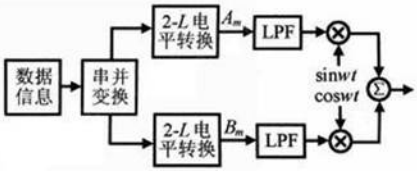


有线前端与接收端的原理框图

下面简单说明有线前端每个部分的功能：
（1）基带接口：该单元将数据结构与信号源格式匹配，帧结构应与包含同步字节的 MPEG-2 TS 流一致。
（2）SYNC1反转和随机化。该单元将依据 MPEG-2 帧结构转换 SYNC1字节；同时为了频谱成形，对数据流进行随机化。
（3）RS编码器。将每一个TS包送入RS编码器进行RS(204,188)信道编码。
（4）卷积交织器。完成深度 I=12 的卷积交织信道编码。
（5）字节变换到 m 比特符号。将字节变换为 QAM 符号。
（6）差分编码。为了获得旋转不变星座图，将每符号两个最高有效位进行差分编码。
（7）基带成形。将差分编码的符号映射到I、Q分量。此外对 I 和 Q 信号进行平方根升余弦滚降滤波。
（8）QAM 调制和物理接口。完成 QAM 调制。之后，它将 QAM 已调制信号连接到有线射频信道。
接收端的功能不再详细叙述。接收端只要按照前端的处理顺序进行逆处理就可以得到基带信号。

2.QAM 信号的调制和解调

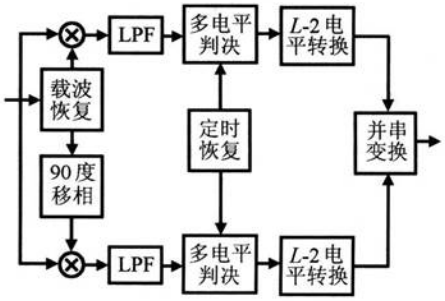
正交振幅调制（QAM）数字传输系统因为其高的频带利用率而被确定为DVB-C 标准。下面简单介绍QAM的调制和解调的步骤。QAM调制的步骤可以由下图表示。



QAM调制原理图

如图所示，输入的码元数据首先经过“串并转换”形成两路数据信号，转换之后，每个码元的持续时间变成原先的2倍；然后经过“2-L电平变换”转换成为多进制的QAM符号；然后将这2路QAM多进制信号分别和2路正交载波相乘；最后再将这2路调制后的信号相加，就形成了调制后的QAM信号。

QAM信号解调的步骤与调制的步骤正好相反，如下图所示。



QAM解调原理图

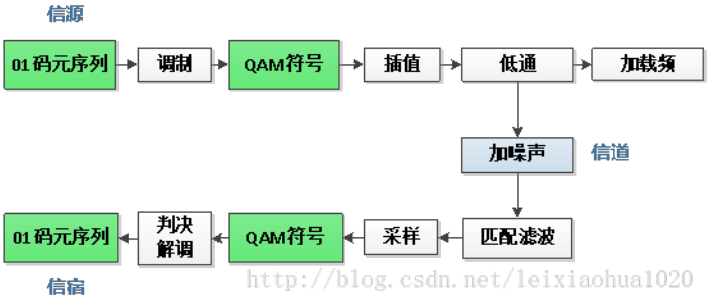
如图所示，QAM信号首先分别使用2路正交的相干载波进行解调；然后经过“多电平判决”电路获得2路多进制的QAM符号；接着经过“L-2电平变换”转换成2路码元数据；最后经过“串并转换”将2路码元数据合并后形成完整的码元数据。

二．仿真

下面介绍仿真实验。本文做了3个仿真实验。第1个实验仿真了16QAM的调制与解调；第2个实验在第一个实验的基础上，仿真了16QAM信号传输过程中信噪比与误码率之间的关系；第3个实验仿真了2FSK,2PSK,4PSK,16PSK,16QAM的信噪比(S/N)与误码率之间的关系对比图。

1. QAM的调制与解调

本实验利用Matlab软件，完成如下图所示的一个基本的基于16QAM的数字通信系统。整个系统可以分成信源、信道和信宿三个部分。信号源产生二进制（0和1）的随机信号，经过调制后形成2路QAM符号（取值-3，-1，1和3）。2路QAM符号分别经过平方根升余弦滤波器，形成等待调制到高频的信号。随后通过给这两路信号添加高斯白噪声的方法，模拟信道的传输环境。然后通过匹配滤波器（平方根升余弦滤波器）。最后经过采样，判决，和解调得到二进制信号。本系统可以从Matlab控制台输出整个过程中关键信号的取值，并且可以在最后统计误码信息。



QAM的调制和解调过程原理图

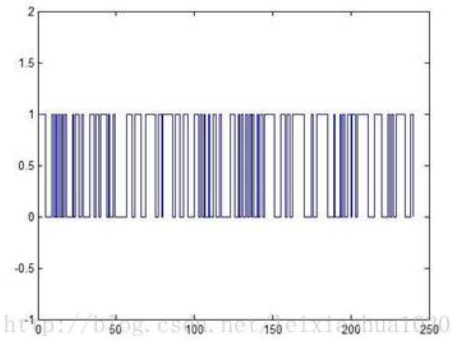
下文将会按照上图所示的信号传递的顺序详细介绍每个步骤。

(a)二进制信号的生成

使用randsrc()函数生成随机二进制序列。本程序默认序列长度为240，其中“1”出现的概率为0.5。代码如下所示。

```
[plain]
1.  %====定义待仿真序列的长度 N
2.  global N
3.  N=240;
4.  %====定义产生‘1’的概率为 p
5.  global p
6.  p=0.5;
7.  %==============================
8.  %首先产生随机二进制序列
9.  source=randsrc(1,N,[1,0;p,1-p]);
10. %画出序列波形
11. h=figure(1);
12. stairs(source);
13. ylim([-1,2]);
14. set(h,'name','随机二进制序列');
```

该步骤运行完成后生成的二进制序列如图所示。



随机二进制序列

(b) 调制

调制步骤将二进制的码元序列转换成为2路QAM符号。该部分的功能在单独定义的modulate_sig()函数中完成。该函数完成了以下几个步骤：

(1) 串并转换。该部分将输入的码元序列中序号为奇数的码元选择出来形成一组数据，然后再将剩下的序号为偶数的码元选择出来形成另一组数据。为了描述方便，下文中称第一组数据称为“I路”，第二组数据称为“Q路”。下文中仅描述I路信号的处理，Q路信号的处理和I路是一模一样的。

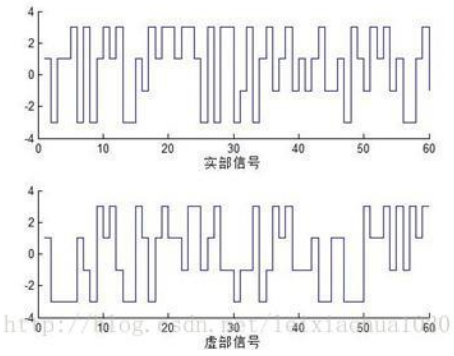
(2) 格雷码映射。将I路数据中每两个码元作为一个单元，按照格雷码的映射规则进行映射。格雷码的映射规则如下表所示。

格雷码映射规则

映射前（二进制码）	映射后（QAM符号）
0 0	-3
0 1	-1
1 1	1
1 0	3

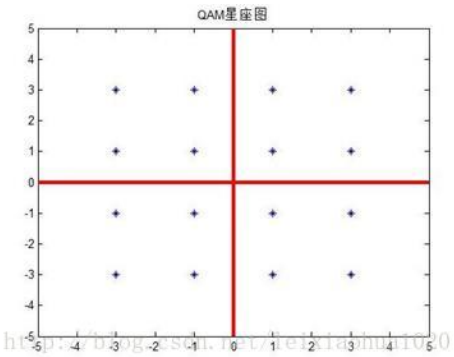
注：上述映射规则可以有个简单的记法：第一个比特位代表符号，“1”代表“+”，第二个比特位代表取值，“1”代表“3”。

经过上述两个步骤处理之后，内容为“0101”的二进制码元序列调制成了内容包含-3，-1，1，3的2路QAM符号。该步骤运行完成后生成的2路QAM符号如下图所示。



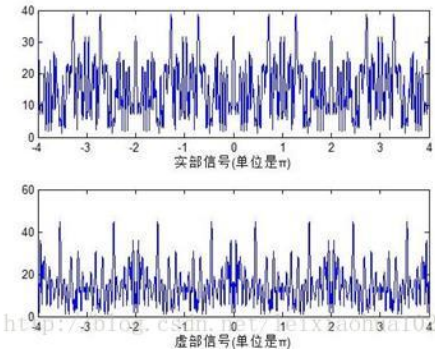
QAM符号序列

将上图中的QAM符号中的I路取值作为x坐标，Q路取值作为y坐标，可以画图得到16QAM符号的“星座图”，如下图所示。



QAM符号星座图

将该QAM符号进行DTFT变换后的频域波形如下图所示。x轴的单位是π。



QAM符号DTFT频域图

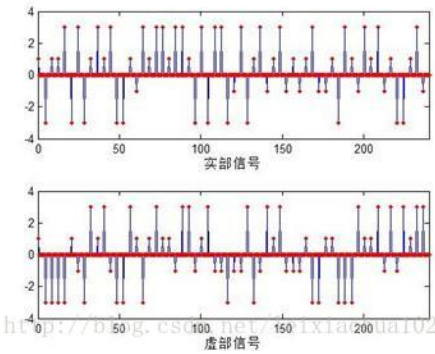
该部分的代码如下所示。

```
[plain]
```

```
%对产生的二进制序列进行QAM调制
function [y1,y2]=modulate_sig(x)
%=====首先进行串并转换，将原二进制序列转换成两路信号
N=length(x);
a=1:2:N;
fprintf('串并转换\n第一路：第1,3,5,7...元素\n第二路：第2,4,6,8...元素\n');
y1=x(a);
y2=x(a+1);
%=====分别对两路信号进行QPSK调制
%======对两路信号分别进行2－4电平变换
a=1:2:N/2;
temp1=y1(a);
temp2=y1(a+1);
y11=temp1*2+temp2;
temp1=y2(a);
temp2=y2(a+1);
y22=temp1*2+temp2;
%=======对两路信号分别进行相位调制
a=1:N/4;
y1=(y11*2-1-4)*1.*cos(2*pi*a);
y2=(y22*2-1-4)*1.*cos(2*pi*a);
%格雷码映射
fprintf('格雷码映射\n');
fprintf('－－－－－－－－－－\n');
fprintf('二进制码  |  格雷码\n');
fprintf(' 00      |  -3\n');
fprintf(' 01      |  -1\n');
fprintf(' 11      |   1\n');
fprintf(' 10      |   3\n');
fprintf('－－－－－－－－－－\n');
fprintf('注：前一位是符号，0代表+，后一位是取值，0代表3；\n');
y1(find(y11==0))=-3;
y1(find(y11==1))=-1;
y1(find(y11==3))=1;
y1(find(y11==2))=3;
y2(find(y22==0))=-3;
y2(find(y22==1))=-1;
y2(find(y22==3))=1;
y2(find(y22==2))=3;
```

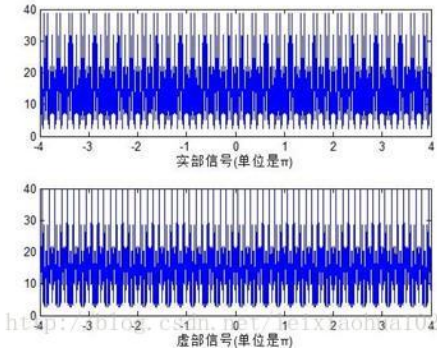
(c)　插值

插值步骤在QAM符号之间插入一些0点。这一步和后面的低通滤波联合起来的作用是模拟信号高频调制之后的频谱。该步骤的实现位于自定义的函数insert_value ()中。具体的方式是分别在信号的I路和Q路中，相邻的两个码字之间添加7个0。经过该步骤处理后，2路QAM符号如下图所示。



差值后两路信号的波形图

上图所示信号进行DTFT变换后的频域波形如下图所示。从图中可以看出，时域信号的差值会造成频域信号的收缩。



插值QAM符号DTFT频域图

该部分的代码如下所示。

```
[plain]
1. %x是待插值的序列，ratio是插值的比例。
2. function y=insert_value(x,ratio)
3. %两路信号进行插值
4. y=zeros(1,ratio*length(x));
5. a=1:ratio:length(y);
6. y(a)=x;<span style="font-family:Arial, Helvetica, sans-serif;"> </span>
```

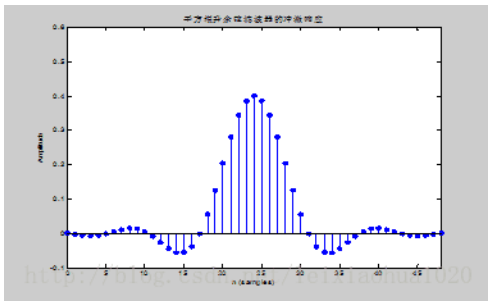
(d) 低通

低通步骤主要用于调整QAM符号的波形，也可以称为“波形成形”。该部分的实现位于自定义的函数rise_cos()中。上述几个步骤中的方波是在本地数字信号处理时常见的波形，但在实际传输时这种方波并不合适，因为使用方波的话会导致相邻传输信号之间的串扰。根据奈奎斯特第一准则，在实际通信系统中一般均使接收波形为升余弦滚降信号。这一过程由发送端的基带成形滤波器和接收端的匹配滤波器两个环节共同实现，因此这两个环节都需要进行平方根升余弦滚降滤波。下面简单介绍平方根余弦滤波器。
平方根升余弦滤波器具有以下定义的理论函数：

$$\begin{cases} H(f)=1 & \text{当} \mid f \mid < f(1-\alpha) \\ H(f)=\left[\frac{1}{2}+\frac{1}{2}\sin\frac{\pi}{2f_N}\left(\frac{f_N-\mid f \mid}{\alpha}\right)\right]^{\frac{1}{2}} & \text{当} f(1-\alpha)\le \mid f \mid \le f(1+\alpha) \\ H(f)=0 & \text{当} \mid f \mid > f(1+\alpha) \end{cases}$$

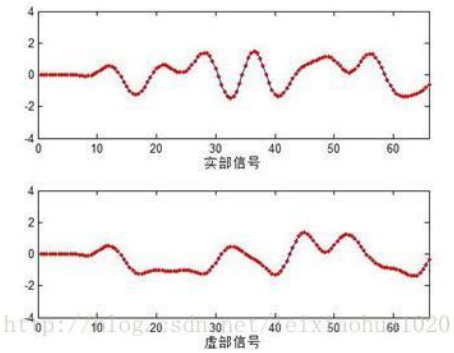
$$f_N=\frac{1}{2T_s}=\frac{R_s}{2}$$

其中： 是奈奎斯特平率， 是滚降系数。
平方根升余弦滤波器的冲击响应如下图所示。



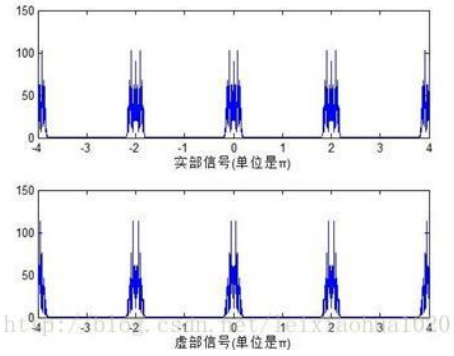
平方根升余弦滤波器的冲激响应曲线

经过该步骤处理后，2路QAM符号如下图所示。



通过低通滤波器后两路信号波形图

上图所示信号进行DTFT变换后的频域波形如下图所示。从图中可以看出，经过低通滤波器之后，频域中高频部分被滤除，保留了低频的信号。



通过低通滤波器后两路信号波形图

该部分的代码如下所示。

```
[plain]
1.  %x1、x2是两路输入信号，fd是信号信息位的频率，fs是信号的采样频率
2.  function [y1,y2]=rise_cos(x1,x2,fd,fs)
3.  %生成平方根升余弦滤波器
4.  [yf, tf]=rcosine(fd,fs, 'fir/sqrt');
5.  %对两路信号进行滤波
6.  [yo1, to1]=rcosflt(x1, fd,fs,'filter/Fs',yf);
7.  [yo2, to2]=rcosflt(x2, fd,fs,'filter/Fs',yf);
8.  y1=yo1;
9.  y2=yo2;<span style="font-family:Arial, Helvetica, sans-serif;"> </span>
```

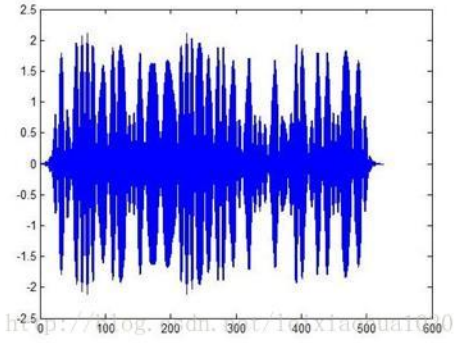
(e) 加载频

加载频在该仿真中属于一个“附加”步骤。该部分主要用于呈现调制到载波之后的波形图，它的实现位于自定义的函数modulate_to_high()中。本程序将通过成形滤波器后的信号调制到10倍于原频率的载波上。由于在仿真的过程中，只能用离散的点来模拟连续信号，因而为了能够显示出一个正弦曲线，至少需要在一个正弦周期内采样到4个以上的点，这里，我们在一个周期内采10个点。假设最初的0、1信号的频率是1Hz，那么I路和Q路符号传输的频率是1/4Hz，而10倍频是建立在I路或Q路符号频率的基础上，也就是说，载频的频率是2.5Hz。调制后的信号就是I路和Q路线性叠加，符合以下公式。

$$y(t) = I(t)\cos 2\pi f_c t - Q(t)\sin 2\pi f_c t$$

其中，fc为载波频率。

该步骤运行后，两路QAM符号合成一路信号，如下图所示。



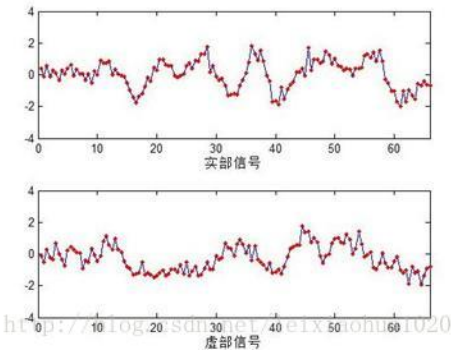
基带信号调制到高频之后的波形图

该部分的代码如下所示。

```
[plain]
1.  %x1,x2代表两路输入信号，f是输入信号的频率，hf是载波的频率
2.  function [t,y]=modulate_to_high(x1,x2,f,hf)
3.  %产生两个中间变量，用来存储插值后的输入信号
4.  yo1=zeros(1,length(x1)*hf/f*10);
5.  yo2=zeros(1,length(x2)*hf/f*10);
6.  n=1:length(yo1);
7.  %对输入信号分别进行插值，相邻的两个点之间加入9个点，且这9个点的值同第0个点的值相同
8.  yo1(n)=x1(floor((n-1)/(hf/f*10))+1);
9.  yo2(n)=x1(floor((n-1)/(hf/f*10))+1);
10. %生成输出输出信号的时间向量
11. t=(1:length(yo1))/hf*f/10;
12. %生成载波调制信号
13. y=yo1.*cos(2*pi*hf*t)-yo2.*sin(2*pi*hf*t);<span style="font-family:Arial, Helvetica, sans-serif;"> </span>
```

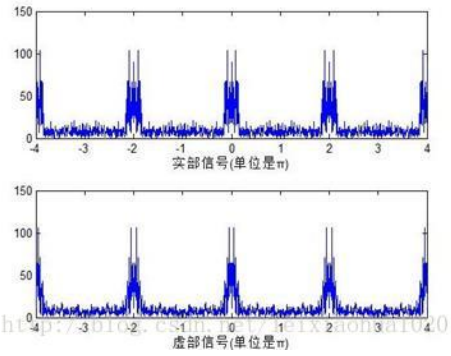
(f) 加噪声

加噪声用于模拟实际的信道中的噪声。本程序设定信道具有高斯白噪声的干扰。该步骤的实现位于自定义的函数generate_noise()中。加入白噪声的2路信号如下图所示。



加入高斯白噪声之后的波形图

上图所示信号进行DTFT变换后的频域波形如下图所示。从图中可以看出，由于白噪声在整个频域上是均匀分布的，所以整个频域范围内都增加了一些噪声。



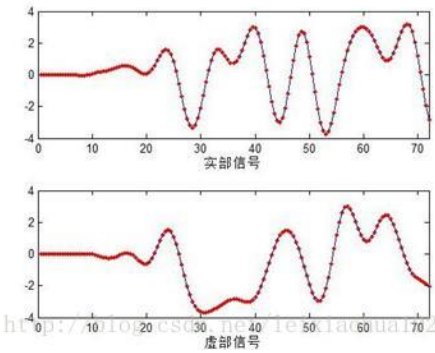
加入高斯白噪声之后DTFT频域图

该部分的代码如下所示。

```
[plain]
1.  %对输入的两路信号加高斯白噪声，返回处理后的两路信号
2.  function [y1,y2]=generate_noise(x1,x2,snr)
3.  %符号信噪比
4.  snr1=snr+10*log10(4);
5.  %所有信号的平均功率
6.  ss=var(x1+i*x2,1);
7.  %加入高斯白噪声
8.  y=awgn([x1+j*x2],snr1+10*log10(ss/10),'measured');
9.  y1=real(y);
10. y2=imag(y);<span style="font-family:Arial, Helvetica, sans-serif;"> </span>
```

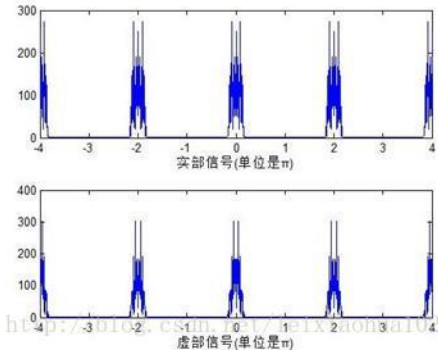
(g) 匹配滤波

匹配滤波用于从信道中接收QAM信号。这一步骤和前面的“低通”步骤是成对出现的。前面步骤的滤波器用于频谱成型，而这一步骤的滤波器主要用于滤除噪声。接收端的匹配滤波与发送端的成形滤波共同实现了数字通信系统的最佳接收。它与成形滤波器共同构成了一个奈奎斯特滤波器。该步骤的实现与“低通”步骤一样，位于generate_noise()函数中。
经过匹配滤波期之后的信号如下图所示。



经过匹配滤波器之后的波形图

上图所示信号进行DTFT变换后的频域波形如下图所示。从图中可以看出，信号高频部分的白噪声在经过低通滤波器之后，已经被消除掉了。



经过匹配滤波器之后DTFT频域图

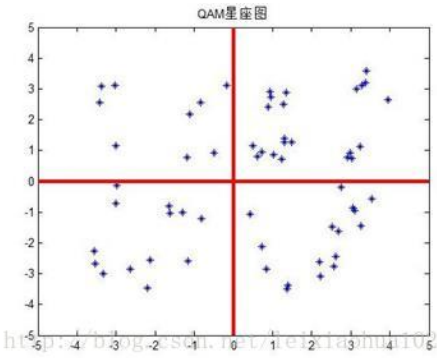
(h) 采样

采样步骤用于还原前面插值步骤处理之前的QAM符号序列。经过前面的插值步骤之后，序列的点数是原本序列的8倍，因此需要去除这些冗余的点。该步骤的具体实现位于自定义的pick_sig()函数中，代码如下所示。

```
[plain]
1. function [y1,y2]=pick_sig(x1,x2,ratio)
2. y1=x1(ratio*3*2+1:ratio:(length(x1)-ratio*3*2));
3. y2=x2(ratio*3*2+1:ratio:(length(x1)-ratio*3*2));
```

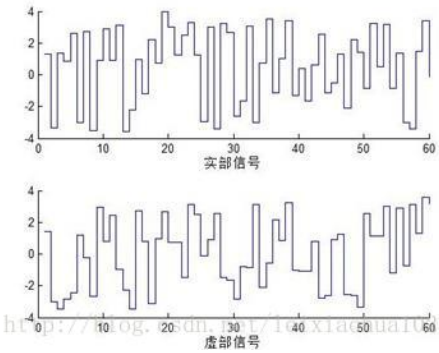
(i) 判决解调

解调步骤用于还原QAM符号序列为二进制码原序列。该步骤的实现位于自定义的demodulate_sig()函数中。经过上述几步骤后，已经可以得到2路QAM16的符号序列，该序列的星座图如下图所示。



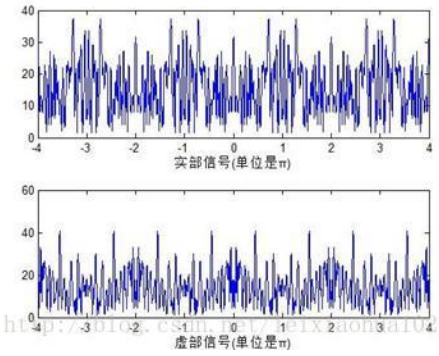
接收的QAM符号星座图

从图中可以看出，原先整齐的星座图经过信道干扰的影响之后之后已经变得有些散乱。这2路QAM符号序列如下图所示。



接收的QAM符号序列

上图所示信号进行DTFT变换后的频域波形如下图所示。从图中可以看出，时域的抽样造成了频域的扩张。



接收的QAM符号序列

对于这样的QAM符号序列，需要做以下几步处理：

(a) 判决。这里可以通过根据最大后验概率（MAP）准则，得到最小检测距离。具体到本仿真中，可以根据下表所示的的规则进行判决。

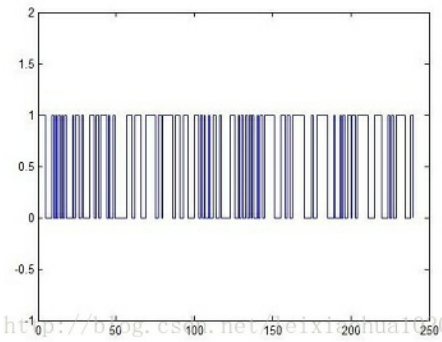
判决电平对应表

判决前的信号的幅度	对应的判决后的幅度
A<-2	－3
-2<=A<=0	－1
0<=A<2	1
A>=2	3

(b) 格雷码映射。按照前文所述的格雷码映射规则，将QAM符号重新转换为二进制码原序列。

(c) 并串转换。将2路二进制码原序列重新合并为1路二进制码原序列。合并的规则和前文所述是一样的，即I路信号作为还原后信号中序号为奇数的码元，Q路则作为还原后信号中序号为偶数的码元。

经过上述步骤后，就可以得到解码后的二进制码元序列了，如下图所示。



恢复的二进制序列

该步骤的代码如下所示。

```plain
function y=demodulate_sig(x1,x2)
%对x1路信号进行判决
xx1(find(x1>=2))=3;
xx1(find((x1<2)&(x1>=0)))=1;
xx1(find((x1>=-2)&(x1<0)))=-1;
xx1(find(x1<-2))=-3;
%对x2路信号进行判决
xx2(find(x2>=2))=3;
xx2(find((x2<2)&(x2>=0)))=1;
xx2(find((x2>=-2)&(x2<0)))=-1;
xx2(find(x2<-2))=-3;
%将x1路信号按格雷码规则还原成0、1信号
temp1=zeros(1,length(xx1)*2);
temp1(find(xx1==-1)*2)=1;
temp1(find(xx1==1)*2-1)=1;
temp1(find(xx1==1)*2)=1;
temp1(find(xx1==3)*2-1)=1;
%将x2路信号按格雷码规则还原成0、1信号
temp2=zeros(1,length(xx2)*2);
temp2(find(xx2==-1)*2)=1;
temp2(find(xx2==1)*2-1)=1;
temp2(find(xx2==1)*2)=1;
temp2(find(xx2==3)*2-1)=1;
%将两路0、1信号合成一路
y=zeros(1,length(temp1)*2);
y(1:2:length(y))=temp1;
y(2:2:length(y))=temp2;
```

(j) 误码率统计

误码率统计步骤用于统计上述整个环节中产生误码的情况。该步骤相对比较简单，使用biterr()函数对比发送端输入的二进制码元序列和接收端得到的二进制码元，就可以得到整个系统的误码率情况。

(k) 程序主干代码

程序主干代码如下所示，位于plot_qam16.m文件中。

```plain
% QAM16的调制与解调示例程序
%
% 雷霄骅
% 中国传媒大学/数字电视技术
% leixiaohua1020@126.com
%
% 本程序演示了QAM的调制与解调过程。
% 程序输出了上述过程中的图形，并且在控制台中输出了相关的信息。
%
%
clear;
%关闭窗口
close all;
%白色背景
set(0,'defaultfigurecolor','w');
%====定义待仿真序列的长度 N
global N
N=240;
%====定义产生‘1’的概率为 p
global p
p=0.5;
%================================
%首先产生随机二进制序列
source=randsrc(1,N,[1,0;p,1-p]);
%画出序列波形
h=figure(1);
stairs(source);
ylim([-1,2]);
set(h,'name','随机二进制序列');
%================================
%对产生的二进制序列进行QAM调制
[source1,source2]=modulate_sig(source);
%================================
%画出序列波形
h=figure(2);
plot_2way_stair(source1,source2);
set(h,'name','QAM符号序列');
%输出坐标
fprintf('\nQAM符号的坐标：\n');
for i=1:length(source1);
    if mod(i,10)==0
        fprintf('\n');
    end
    fprintf('(%.3f,%.3f);',source1(i),source2(i));
end;
fprintf('\n');
%================================
%画出星座图
h=figure(3);
```

```
plot_astrology(source1,source2);
set(h,'name','QAM符号星座图');
%================================
%画出频域图
h=figure(4);
plot_2way_f_dtft(source1,source2);
set(h,'name','QAM符号DTFT频域图');
%plot_2way_f_dft(source1,source2,50);
%set(h,'name','QAM符号DFT频域图');
%================================
%两路信号进行插值
sig_insert1=insert_value(source1,8);
sig_insert2=insert_value(source2,8);
%================================
%画出两路信号的波形图
h=figure(5);
plot_2way(sig_insert1,sig_insert2,length(sig_insert1),0.5);
set(h,'name','插值后两路信号的波形图');
%================================
%画出频域图
h=figure(6);
plot_2way_f_dtft(sig_insert1,sig_insert2);
set(h,'name','插值QAM符号DTFT频域图');
%plot_2way_f_dft(sig_insert1,sig_insert2,50);
%set(h,'name','插值QAM符号DFT频域图');
%================================
%通过低通滤波器
[sig_rcos1,sig_rcos2]=rise_cos(sig_insert1,sig_insert2,0.25,2);
%================================
%画出两路信号的波形图
h=figure(7);
plot_2way(sig_rcos1,sig_rcos2,length(sig_rcos1)/4,0.5);
set(h,'name','通过低通滤波器后两路信号波形图');
%================================
%画出频域图
h=figure(8);
plot_2way_f_dtft(sig_rcos1',sig_rcos2');
set(h,'name','通过低通滤波器后DTFT频域图');
%================================
%====将基带信号调制到高频上
[t,sig_modulate]=modulate_to_high(sig_rcos1,sig_rcos2,0.25,2.5);
h=figure(9);
%plot(t(1:500),sig_modulate(1:500));
plot(t,sig_modulate);
set(h,'name','基带信号调制到高频之后的波形图');
%================================

%====加入高斯白噪声
snr=10;
[x1,x2]=generate_noise(sig_rcos1,sig_rcos2,snr);
sig_noise1=x1';
sig_noise2=x2';
h=figure(10);
plot_2way(sig_noise1,sig_noise2,length(sig_noise1)/4,0.5);
set(h,'name','加入高斯白噪声之后的波形图');
%================================
%画出频域图
h=figure(11);
plot_2way_f_dtft(sig_noise1,sig_noise2);
set(h,'name','加入高斯白噪声之后DTFT频域图');

%====经过匹配滤波器
[sig_match1,sig_match2]=rise_cos(sig_noise1,sig_noise2,0.25,2);
h=figure(12);
plot_2way(sig_match1,sig_match2,length(sig_match1)/4,0.5);
set(h,'name','经过匹配滤波器之后的波形图');
%================================
%画出频域图
h=figure(13);
plot_2way_f_dtft(sig_match1',sig_match2');
set(h,'name','经过匹配滤波器之后DTFT频域图');
%采样
[x1,x2]=pick_sig(sig_match1,sig_match2,8);
sig_pick1=x1;
sig_pick2=x2;
%画出星座图
h=figure(14);
plot_astrology(sig_pick1,sig_pick2);
set(h,'name','接收的QAM符号星座图');
%画出序列波形
h=figure(15);
plot_2way_stair(sig_pick1,sig_pick2);
set(h,'name','接收的QAM符号序列');
%输出坐标
fprintf('\nQAM符号的坐标（接收）：\n');
for i=1:length(source1);
    if mod(i,10)==0
        fprintf('\n');
    end
    fprintf('(%.3f,%.3f);',sig_pick1(i),sig_pick2(i));
end;
```

```
fprintf('\n');
%================================
%画出频域图
h=figure(16);
plot_2way_f_dtft(sig_pick1',sig_pick2');
set(h,'name','接收的QAM符号序列DTFT频域图');
%解调
signal=demodulate_sig(sig_pick1,sig_pick2);
h=figure(17);
stairs(signal);
ylim([-1,2]);
set(h,'name','恢复的二进制序列');
%误码率
[number,Pe] = biterr(source,signal);
fprintf('误码数以及误码率：\n');
fprintf('%d,%f;\n',number,Pe);
hold on
```

2. QAM传输过程中信噪比与误码率之间的关系

本实验在前一个实验的基础上，仿真了16QAM传输过程中信噪比与误码率之间的关系。本仿真的大部分代码与前一个实验是一样的，主要的不同点在于它计算了QAM理论的误码率，并且与实际的误码率进行了对比，整个程序的流程简单描述如下：

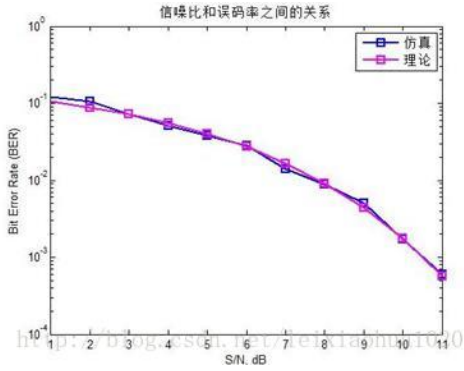
(a) 设定信噪比

(b) 计算16QAM该信噪比下的误码率

(c) 仿真16QAM该信噪比下的误码率

(d) 重新设定信噪比，重复（2）（3）步骤

该仿真实验选择了1-11dB的信噪比对16QAM进行了仿真，仿真的结果如下图所示。



16QAM信噪比与误码率之间的关系的仿真结果

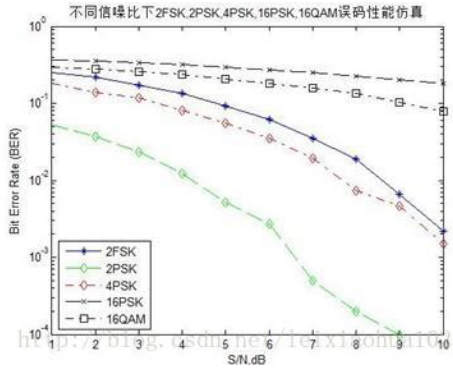
从仿真结果图来看，仿真的误码率十分贴近理论的误码率。程序的代码位于plot_qam16_ber.m文件中，代码的内容和前一个实验是类似的，关键代码大致如下。

```plain
% QAM16的信噪比(S/N)与误码率关系图
%
% 雷霄骅
% 中国传媒大学/数字电视技术
% leixiaohua1020@126.com
%
% 本程序可以得到QAM调制中信噪比(S/N)与误码率之间的关系图
%
%
clear;
close all;
%白色背景
set(0,'defaultfigurecolor','w');
%用来仿真QAM的误bit率
snr=1:1:11;
%先来计算理论误bit率
error_theory=(1-(1-(2*(1-1/sqrt(16))*1/2*erfc(1/sqrt(2)*sqrt(3*4*10.^(snr/10)/(16-1))))).^2)/4;
%用理论的误bit率来决定需要仿真的点数
N=floor(1./error_theory)*100+100;
%最少5000点
N(find(N<5000))=5000;

for i=1:length(N);
        %一次仿真，代码略
end
figure(4);
%画出图形
semilogy(snr,error_bit,'bs-','LineWidth',2);
hold on
semilogy(snr,error_theory,'ms-','LineWidth',2)
legend('仿真','理论');
xlabel('S/N, dB');
ylabel('Bit Error Rate (BER)');
title('信噪比和误码率之间的关系');
```

3. 2FSK,2PSK,4PSK,16PSK,16QAM的信噪比与误码率之间的关系对比

上个实验仿真了16QAM传输过程中信噪比与误码率之间的关系。为了能和其他调制方式做一个对比，本实验仿真了2FSK，2PSK，4PSK，16PSK，16QAM传输过程中信噪比与误码率之间的关系。与前两个实验不同之处在于，本实验没有使用自行编写的程序完成这几种调制方式的仿真，而是使用Matlab的接口函数完成了仿真，实验的结果如下图所示。



2FSK,2PSK,4PSK,16PSK,16QAM的信噪比与误码率之间的关系对比的仿真结果

从图中可以看出，在相同信噪比的前提下，不同调制技术之间的误码率关系如下所示：

2PSK < 4PSK < 2FSK < 16QAM < 16PSK

程序的源代码如下。

```plain
% 2FSK,2PSK,4PSK,16PSK,16QAM的
% 信噪比(S/N)与误码率之间的关系对比图
%
% 雷霄骅
% 中国传媒大学/数字电视技术
% leixiaohua1020@126.com
%
% 本程序可以计算得到2FSK,2PSK,4PSK,16PSK,16QAM信噪比
% 与误码率之间的关系图
%
%
clear;
close all;
%白色背景
set(0,'defaultfigurecolor','w');
M2=2;
M4=4;
M16=16;
k2=log2(M2);
k4=log2(M4);
k16=log2(M16);


Fs = 16;
nsamp = 17;
freq_sep = 8;
x = randint(10000,1);
%mod
y_2fsk = fskmod(x,M2,freq_sep,nsamp,Fs);
y_2psk = pskmod(x,M2);
x4 = bi2de(reshape(x,k4,length(x)/k4).','left-msb');
y_4psk = pskmod(x4,4);
x16 = bi2de(reshape(x,k16,length(x)/k16).','left-msb');
y_16psk = pskmod(x16,16);
y_16qam = qammod(x16,16);

N=10;
for j=1:N
    SNR=j;
    %[],'dB'
    y_2fsk_noise = awgn(y_2fsk,SNR-10*log10(Fs),'measured');
    y_2psk_noise = awgn(y_2psk,SNR,'measured');
    y_4psk_noise = awgn(y_4psk,SNR,'measured');
    y_16psk_noise = awgn(y_16psk,SNR,'measured');
    y_16qam_noise = awgn(y_16qam,SNR,'measured');
    %demod
    z_2fsk = fskdemod(y_2fsk_noise,M2,freq_sep,nsamp,Fs);
    z_2psk = pskdemod(y_2psk_noise,M2);
    z_4psk = pskdemod(y_4psk_noise,M4);
    z_16psk = pskdemod(y_16psk_noise,M16);
    z_16qam = qamdemod(y_16qam_noise,M16);
    %x = [1 1 1 1 1];
    %y = [0 0 0 0 1];
    %[number,ratio]=biterr(x,y)得到 number=4, ratio=0.8
    [num_2fsk(j),ber_2fsk(j)] = biterr(x,z_2fsk);
    [num_2psk(j),ber_2psk(j)] = biterr(x,z_2psk);
    [num_4psk(j),ber_4psk(j)] = biterr(x4,z_4psk);
    [num_16psk(j),ber_16psk(j)] = biterr(x16,z_16psk);
    [num_16qam(j),ber_16qam(j)] = biterr(x16,z_16qam);
end

%theoryBer = (1/k)*3/2*erfc(sqrt(k*0.1*(10.^(EbNo/10))));
%semilogy(EbNo,theoryBer,'ms-');
%hold on;
semilogy(1:N,ber_2fsk,'-*b');
hold on;
semilogy(1:N,ber_2psk,'--dg');
hold on;
semilogy(1:N,ber_4psk,'-.dr');
hold on;
semilogy(1:N,ber_16psk,'--xk');
hold on;
semilogy(1:N,ber_16qam,'-.sk');
title('不同信噪比下2FSK,2PSK,4PSK,16PSK,16QAM误码性能仿真');
legend('2FSK','2PSK','4PSK','16PSK','16QAM',3);
xlabel('S/N,dB');
ylabel('Bit Error Rate (BER)')
hold off
```

三．下载

QAM调制与解调仿真（Matlab程序）

CSDN下载地址： http://download.csdn.net/detail/leixiaohua1020/8453115

文章标签： QAM 解调 调制 DVB-C 滤波

个人分类： 数字电视网络