第三次编程作业

无 36　　李思涵　　2013011187　　lisihan969@gmail.com

2015 年 10 月 31 日

目录

1 编程题 1

1.1 产生一个长为 1000 的二进制随机序列，“0” 的概率为 0.8，“1” 的概率为 0.2

为了产生特定分布的二进制随机序列，我们先使用 rand 函数产生 0 到 1 间的随机数，然后将数值小于等于 0 的概率的数置成 0，其他数置成 1。代码如下所示。

```
LEN = 1000;
FS = 8;
WIDTH = 0.5;
P_ZERO = 0.8;
PLOT_LEN = 20;

signal = rand(LEN, 1);
signal(signal <= P_ZERO) = 0;
signal(signal ~= 0) = 1;
```

1.2 对上述数据进行归零 AMI 编码，脉冲宽度为符号宽度的百分之五十，波形采样率为符号率的 8 倍。画出前 20 个符号对应的波形（同时给出前 20 位信源序列）

我们首先来实现 ami 函数，其作用是对信号进行 AMI 编码。

我们的思路十分简单，首先找到信号中的所有 1，然后将 $(-1)^k$ 序列填充进去。具体代码如下：

```
function code = ami(signal, fs, width)
    signal(signal == 1) = (-1).^(0:sum(signal)-1);
    code = zeros(length(signal) * fs, 1);

    pad = round(fs / 2 * (1 - width));

    for n = 1:length(code)
        index = ceil(n / fs);
        pos = mod(n - 1, fs);

        if pos >= pad && pos < fs - pad
            code(n) = signal(index);
        else
            code(n) = 0;
        end
    end
end
```

为了方便，我们在这里同时给出了三种编码方式的编码和波形绘制。

```
code_ami = ami(signal, FS, WIDTH);
code_hdb3 = hdb3(signal, FS, WIDTH);
code_miller = miller(signal, FS);

t = linspace(0, PLOT_LEN - 1 - 1/FS, PLOT_LEN * FS);

sample_signal = repmat(signal', FS, 1);
sample_signal = sample_signal(:);

figure
subplot 411
stem(t, sample_signal(1:PLOT_LEN*FS));
title Signal
subplot 412
stem(t, code_ami(1:PLOT_LEN*FS));
title AMI
subplot 413
stem(t, code_hdb3(1:PLOT_LEN*FS));
title HDB3
subplot 414
stem(t, code_miller(1:PLOT_LEN*FS));
title Miller
xlabel t
```

得到波形如图所示。

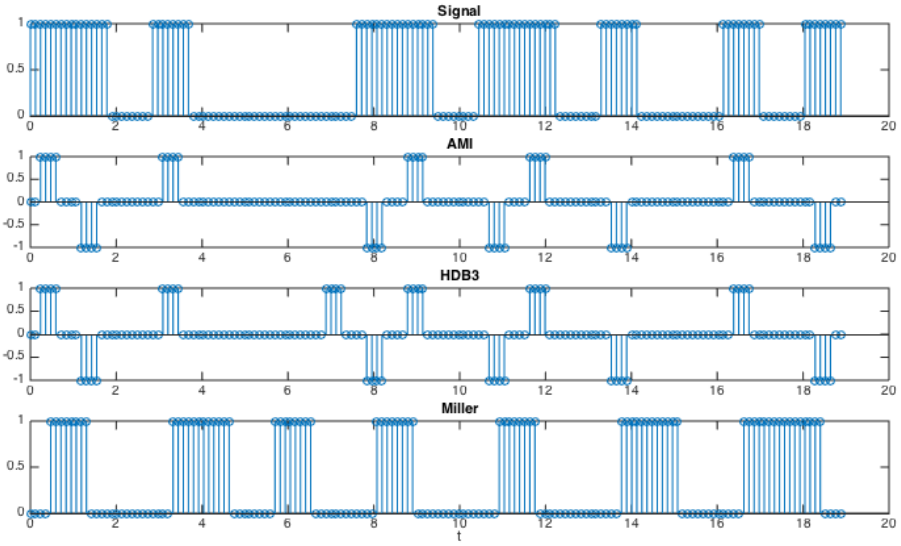

图 1: 编码波形

1.3 改用 HDB3 码，画出前 20 个符号对应的波形

同样的，让我们先实现 HDB3 编码器。我们的方法是，跟踪上一个非零波形的符号，连续 0 的数目以及上一个 V 到现在的 B 的数目。然后我们对信号序列进行遍历，用状态机的思想判断每一个点的输出波形。具体代码如下。

```
function code = hdb3(signal, fs, width)
    s = zeros(length(signal), 1);

    last_sign = -1;
    zero_count = 0;
    b_count = 0;
    for n = 1:length(signal)
        if signal(n) ~= 0  % B.
            last_sign = -last_sign;
            s(n) = last_sign;
            zero_count = 0;
            b_count = b_count + 1;
        else
            zero_count = zero_count + 1;
            if zero_count == 4
                if mod(b_count, 2) == 0
                    last_sign = -last_sign;
                    s(n - 3) = last_sign;  % Add a B.
                end
                s(n) = last_sign;  % Add a V.
                zero_count = 0;
                b_count = 0;
            end
        end
    end

    code = zeros(length(s) * fs, 1);
    pad = round(fs / 2 * (1 - width));

    for n = 1:length(code)
        index = ceil(n / fs);
        pos = mod(n - 1, fs);

        if pos >= pad && pos < fs - pad
            code(n) = s(index);
        else
            code(n) = 0;
        end
```

```
    end
end
```

最后的到的波形如波形图中所示。

1.4 改用密勒码，画出前 20 个符号对应的波形

同样的，我们先来实现 `miller` 函数。我们直接对整个信号序列进行遍历，根据当前点和前一点的取值来决定当前点的波形输出。代码如下所示。

```
function code = miller(signal, fs)
    s = zeros(length(signal) * 2, 1);

    last_signal = 0;
    last_s = 0;
    for n = 1:length(signal)
        if signal(n) == 1
            delta = [0 1];
        elseif last_signal == 1
            delta = [0 0];
        else
            delta = [1 1];
        end

        s(2*n-1:2*n) = mod(delta + last_s, 2);
        last_signal = signal(n);
        last_s = s(2 * n);
    end

    code = repmat(s', round(fs / 2), 1);
    code = code(:);
end
```

最后的到的波形如波形图所示。

1.5 分别对上述 1000 个符号的波形进行功率谱估计，画出功率谱

我们直接对得到的波形序列调用 `psd` 函数，进行功率谱的估计。代码如下：

```
figure
subplot 411
psd(sample_signal);
title Signal
subplot 412
psd(code_ami);
title AMI
```

```
subplot 413
psd(code_hdb3);
title HDB3
subplot 414
psd(code_miller);
title Miller
xlabel t
```

得到原信号和三个编码波形的功率谱如图所示。

需要注意的是，由于我们取的是 8 被采样率，而 psd 函数对是波形默认画的是单边功率谱，故我们在看功率谱的时候应该取其 1/4 进行观察。可以看到，这些信号的波形与书中描述的基本一致。

1.6 改变信源 “0” 的概率，观察 AMI 码的功率谱变化情况

我们去信源为 0 的概率分别为 0, 0.01, 0.1, 0.5, 0.9, 0.99, 1 进行试验，生成相应的随机信号。代码如下：

```
figure
hold on

for p_zero = [0, 0.01, 0.1, 0.5, 0.9, 0.99, 1]
    signal = rand(LEN, 1);
    signal(signal <= p_zero) = 0;
    signal(signal ~= 0) = 1;
    psd(ami(signal, FS, WIDTH));
end

legend('p=0', 'p=0.01', 'p=0.1', 'p=0.5', ...
       'p=0.9', 'p=0.99', 'p=1');
```

得到不同概率下功率谱的曲线如图所示。

因为和上面一样的原因，我们在看这张功率谱的时候应该只看前 1/4。

从图中我们可以看到，随着信源 0 的概率的增大，功率谱会变得越来越平滑。这其实不难理解，因为在 AMI 编码中，0 会被编码成 0，而 1 会编码成 1 或 -1，故 0 越多，信号就越接近于冲激函数，功率谱便越接近于恒定值，即越平滑。

需要注意的是，当 0 的概率为 1 时，功率谱实际上为恒 0，所以在图上没有表示出来。

2 编程题 2

2.1 产生长度为 100 的随机二进制序列

为了方便，我们在这里使用 randi 生成等概的二进制序列，即 0 和 1 的概率分别为 50%。

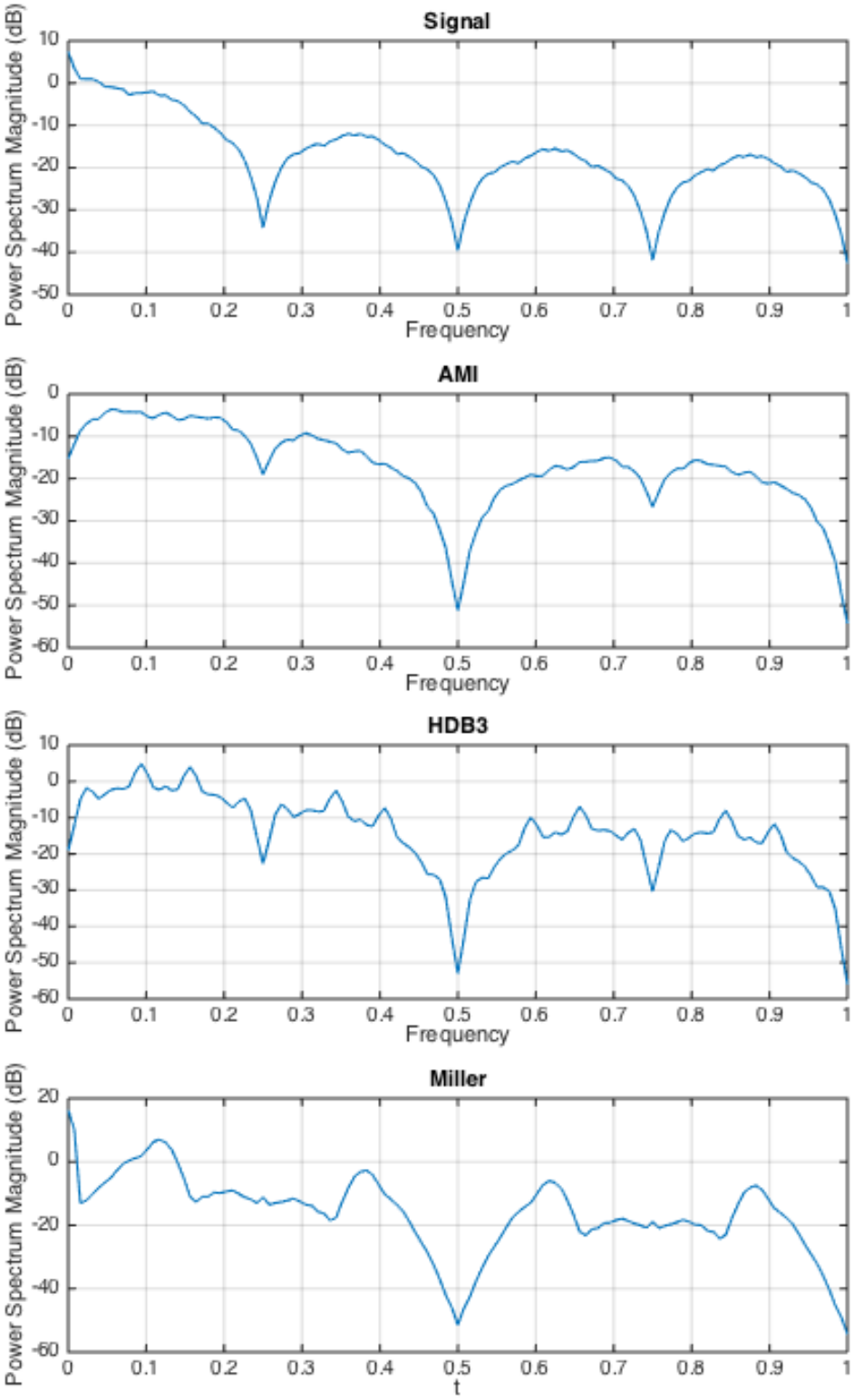
Signal
Power Spectrum Magnitude (dB)
Frequency
AMI
Power Spectrum Magnitude (dB)
Frequency
HDB3
Power Spectrum Magnitude (dB)
Frequency
Miller
Power Spectrum Magnitude (dB)
t

图 2: 波形功率谱

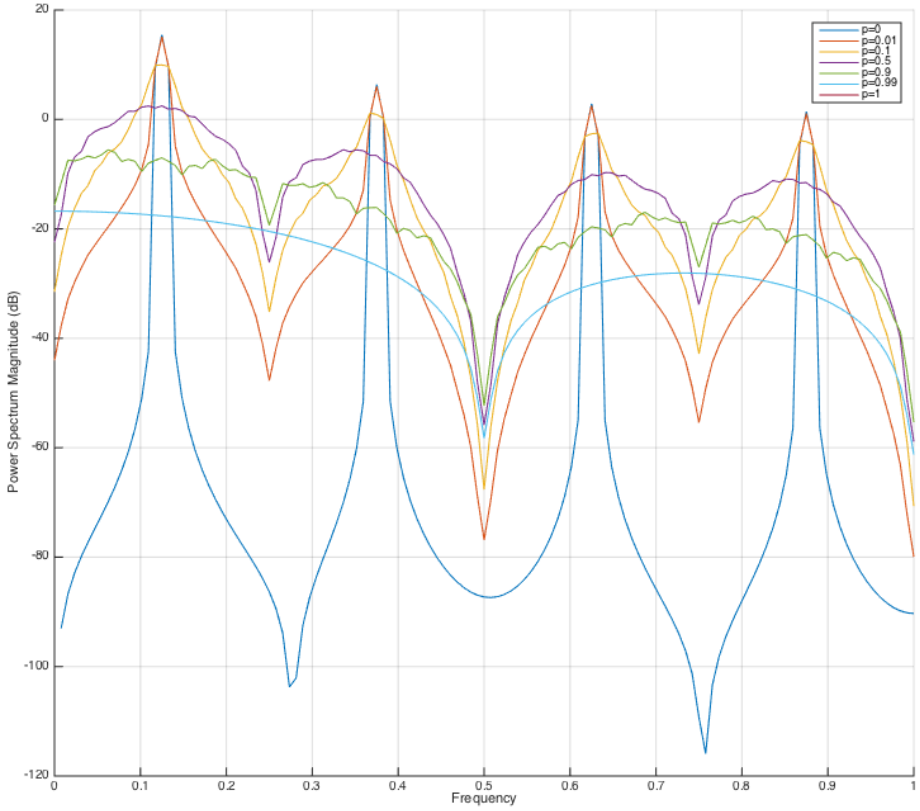

图 3: 不同概率下 AMI 编码功率谱曲线

```
LEN = 100;
F = 10;
F_RECEIVE = 10.05;
FS = 100;
PLOT_LEN = 10;
PLOT_LEN_RECEIVE = 20;

signal = randi([0 1], LEN, 1);
```

2.2 发送载波频率为 10 倍比特率，画出过采样率为 100 倍符号率的 BPSK 调制波形（前 10 个比特）及其功率谱

我们先来编写 bpsk 函数，用于 BPSK 编码。

```
function [code, t] = bpsk(signal, f, fs)
    t = linspace(0, length(signal) - 1/fs, length(signal) * fs)';
    code = (2 * signal(int32(floor(t) + 1)) - 1) .* ...
           cos(2 * pi * f * t - pi / 2);
```

然后，我们画出波形图和功率谱。

```
[code t] = bpsk(signal, F, FS);

sample_signal = repmat(signal(1:PLOT_LEN)', FS, 1);
sample_signal = sample_signal(:);

range = 1:PLOT_LEN*FS;
range_receive = 1:PLOT_LEN_RECEIVE*FS;
range2 = 1:(PLOT_LEN+1)*FS;  % For DPSK.

% First bits.
figure
subplot 211
plot(t(range), sample_signal(range));
title Signal
subplot 212
plot(t(range), code(range));
title BPSK
xlabel t

% PSD.
figure
psd(code);
```

可以看到，BPSK 调制后的波形在原频率处有一个峰值。这是因为调制后，信号的频率分量仍然是以载波信号的频率为主，故功率谱也在此频率上有峰值。

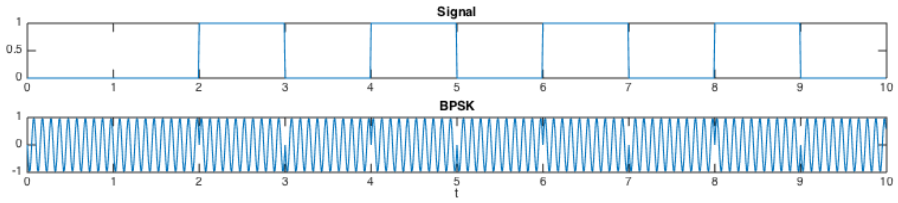

图 4: BPSK 调制波形

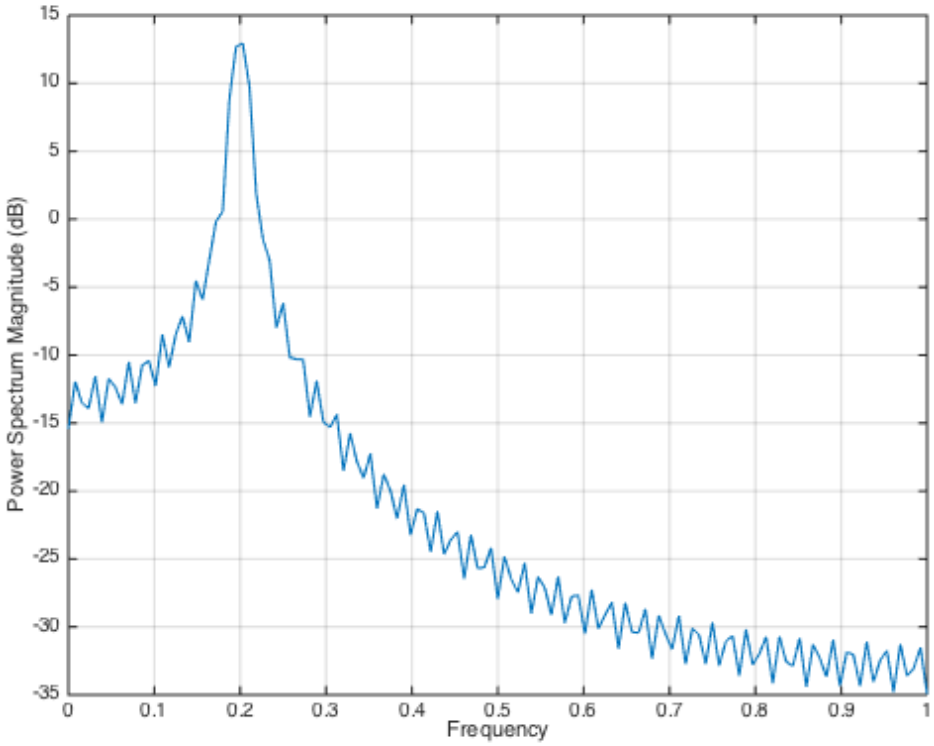

图 5: BPSK 功率谱

2.3 相干解调时假设收发频率相位相同，画出 x(t) 的波形。假设低通滤波器的冲激响应为连续 10 个 1（其余为 0），或连续 12 个 1（其余为 0），分别画出两种滤波器下的 y(t)，及判决输出 (前 10 个比特)

我们使用 `filter` 函数来进行低通滤波，具体代码如下：

```
% F.
x = code .* cos(2 * pi * F * t + pi / 2);
y1 = filter(ones(1, 10), 1, x);
y2 = filter(ones(1, 12), 1, x);

% Sample & decode.
samples = floor(FS / 2):FS:length(t);
decoded1 = y1(samples) >= 0;
decoded2 = y2(samples) >= 0;

figure
subplot 511
plot(t(range), x(range));
title x(t)
subplot 512
plot(t(range), y1(range));
title 'y(t), 10 ones'
subplot 513
plot(t(range), y2(range));
title 'y(t), 12 ones'
subplot 514
stem(decoded1(1:PLOT_LEN));
axis([0.5 10.5 0 1])
title 'decoded, 10 ones'
subplot 515
stem(decoded2(1:PLOT_LEN));
axis([0.5, PLOT_LEN + 0.5, 0 1])
title 'decoded, 12 ones'
```

可以看到，当低通滤波器的冲激响应为 10 个 1 时，得到的波形十分光滑。而当冲激响应为 12 个 1 时，得到的波形有小范围的波动。这是因为，载波的周期的是个采样间隔。当冲激响应中 1 的个数与载波的频率相匹配时（即为整数倍时），载波便对滤波器的输出没有影响。相反的话，则会造成部分载波通过滤波器，使得到的波形有震荡。

2.4 接收载波频率为 10.05 倍比特率，初相位相同。画出 x(t) 的波形。假设低通滤波器的冲激响应为连续 10 个 1，画出两种滤波器下的 y(t)，及判决输出（前 20 个比特）

我们使用 10.05 倍频率的接受载波频率进行解调，得到结果如下。

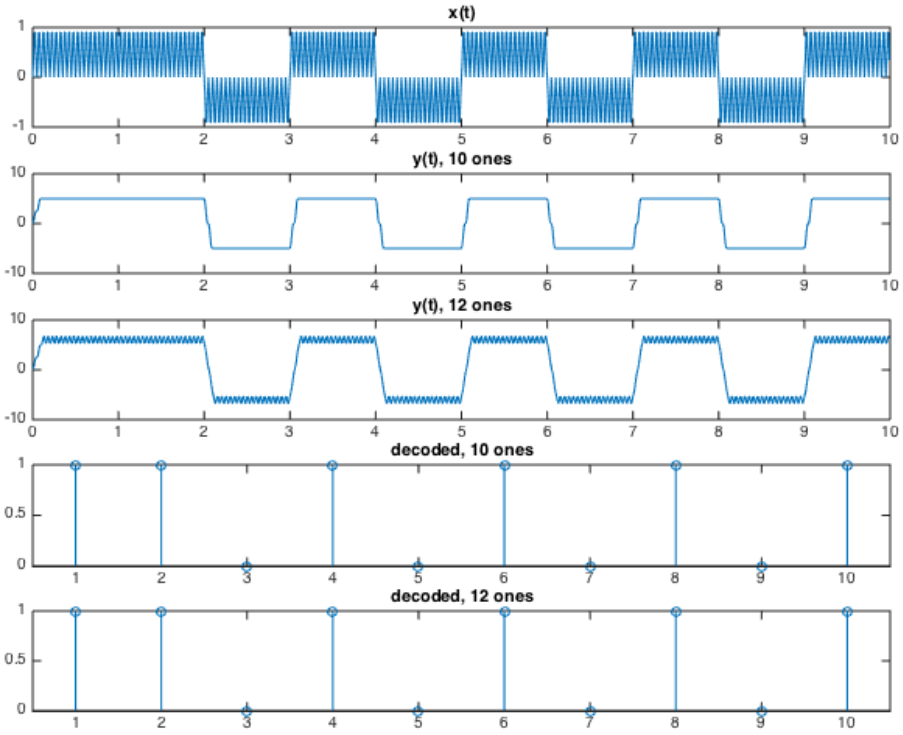

图 6: BPSK 解码

```
% F_RECEIVE.
x_receive = code .* cos(2 * pi * F_RECEIVE * t + pi / 2);
y_receive = filter(ones(1, 10), 1, x_receive);

% Sample & decode.
decoded_receive = y_receive(samples) >= 0;

figure
subplot 311
plot(t(range_receive), x_receive(range_receive));
title 'x(t), F_s = 10.05'
subplot 312
plot(t(range_receive), y_receive(range_receive));
title 'y(t), F_s = 10.05'
subplot 313
stem(decoded_receive(1:PLOT_LEN_RECEIVE));
axis([0.5, PLOT_LEN_RECEIVE + 0.5, 0 1])
title 'decoded, F_s = 10.05'
```

从图中可以看到，由于接收发送载波频率不匹配，这一偏差相当于对解调得到信号的幅度进行了调制。从第二幅图中可以看出，解调得到的 1 和 0 的波形分别在正弦曲线上。所

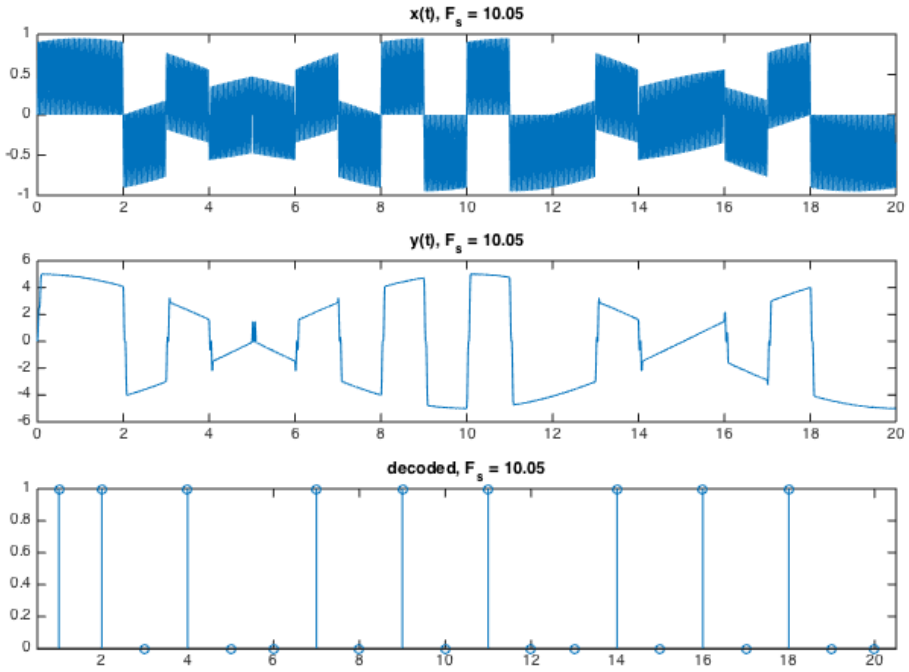

图 7: 载波频率为 10.05 倍比特率时 BPSK 解码

以，在二者幅度都很小的时候，很容易出现误判；在二者反相的情况下，则一定会出现误判。

2.5 采用 DPSK 及延时差分相干解调，载波频率为 10 倍比特率，画出 a, b, c, d 点的波形（前 10 个比特）

我们先来实现 DPSK 编码：

```
function [code, t] = dpsk(signal, f, fs)
    % We need (length(signal) + 1) T.
    len = length(signal) + 1;
    t = linspace(0, len - 1/fs, len * fs)';
    delta = [0; mod(cumsum(signal), 2)];
    code = (2 * delta(int32(floor(t) + 1)) - 1) .* ...
           cos(2 * pi * f * t - pi / 2);
```

然后，我们对 DPSK 调制后的信号进行延时差分解调。

```
% DPSK
[code2, t2] = dpsk(signal, F, FS);

pad = zeros(FS, 1);
a = [code2; pad];
b = [pad; code2];
c = a .* b;
```

```
d = filter(ones(1, 10), 1, c);

figure
subplot 411
plot(t2(range2), a(range2));
title a
subplot 412
plot(t2(range2), b(range2));
title b
subplot 413
plot(t2(range2), c(range2));
title c
subplot 414
plot(t2(range2), d(range2));
title d
```

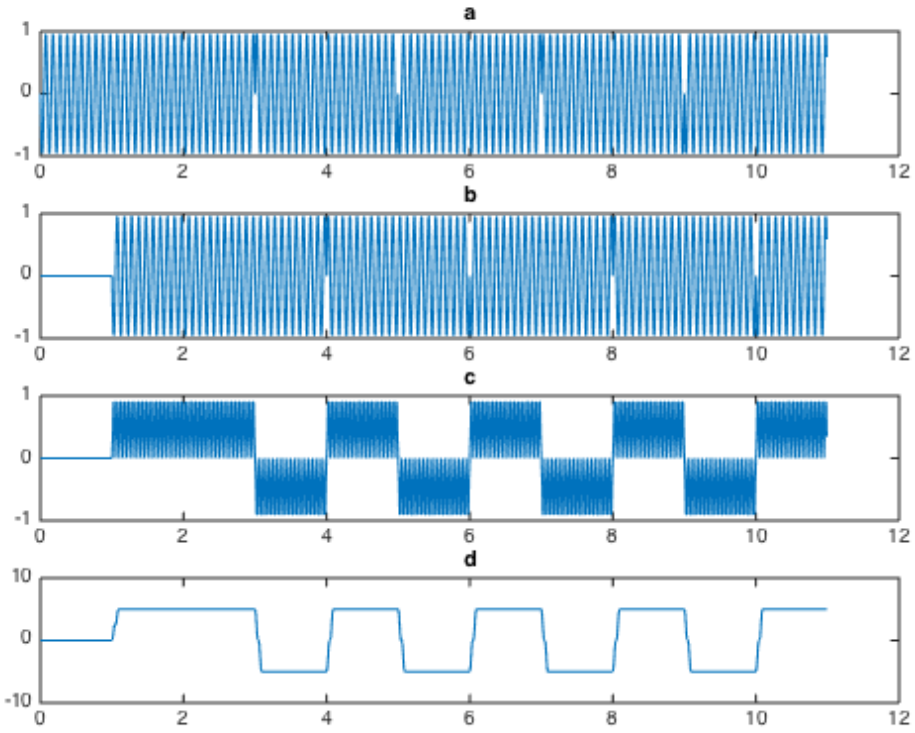

图 8: 载波频率为 10 倍比特率时 DPSK 波形

可以看到，我们成功恢复出了原信号。

2.6 DPSK 及延时差分相干解调，载波频率为 10.25 倍比特率时，画出 a, b, c, d 点的波形（前 10 个比特）

当载波频率为 10.25 倍比特率时，b 延迟 100 个采样周期将会导致 1/4 个周期落后，故 a 与 b 相乘时，二者相位相差 /4，得到的信号 c 仍然是一个正弦信号。故低通滤波后基本没有能量剩余，基本上什么都恢复不出来。

```
% DPSK, 10.25
[code2, t2] = dpsk(signal, 10.25, FS);

pad = zeros(FS, 1);
a = [code2; pad];
b = [pad; code2];
c = a .* b;
d = filter(ones(1, 10), 1, c);

figure
subplot 411
plot(t2(range2), a(range2));
title 'a, F_s = 10.25'
subplot 412
plot(t2(range2), b(range2));
title 'b, F_s = 10.25'
subplot 413
plot(t2(range2), c(range2));
title 'c, F_s = 10.25'
subplot 414
plot(t2(range2), d(range2));
title 'd, F_s = 10.25'
```

从图中可以看到，d 信号确实和我们分析的结果一样几乎没有能量剩余。

2.7 DPSK 及延时差分相干解调，载波频率为 10.5 倍比特率时，画出 a, b, c, d 点的波形（前 10 个比特）

当载波频率为 10.5 倍比特率时，b 延迟 100 个采样周期将会导致 1/2 个周期落后，故 a 与 b 相乘时，得到的 c 将与原 c 恰好反相，故最后解调出的信号恰好是原信号按位取反。

```
% DPSK, 10.5
[code2, t2] = dpsk(signal, 10.5, FS);

pad = zeros(FS, 1);
a = [code2; pad];
b = [pad; code2];
c = a .* b;
```

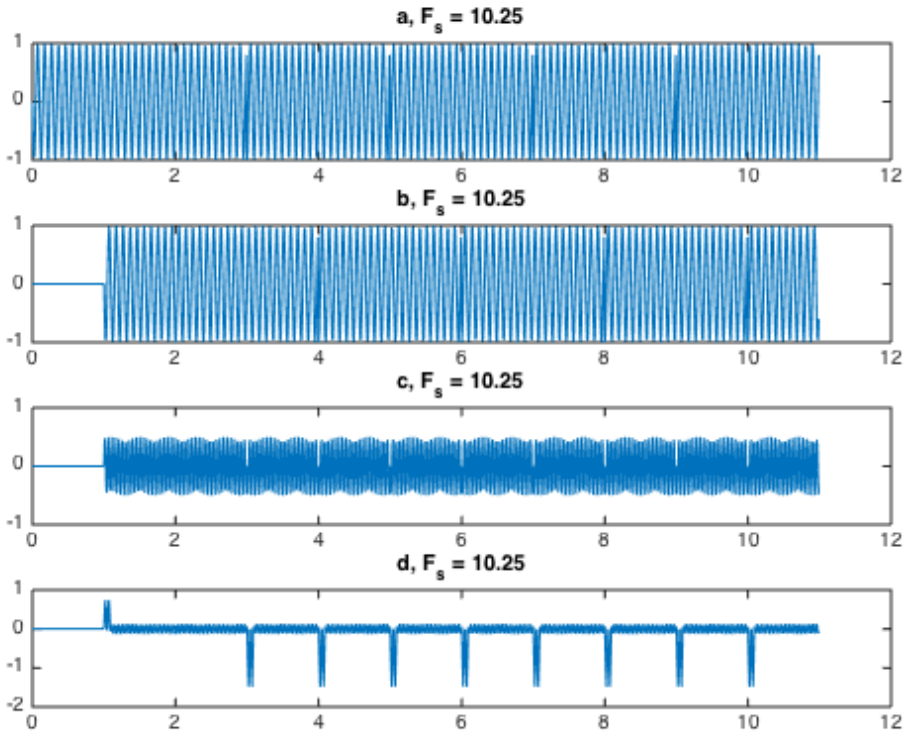

图 9: 载波频率为 10.25 倍比特率时 DPSK 波形

```
d = filter(ones(1, 10), 1, c);

figure
subplot 411
plot(t2(range2), a(range2));
title 'a, F_s = 10.5'
subplot 412
plot(t2(range2), b(range2));
title 'b, F_s = 10.5'
subplot 413
plot(t2(range2), c(range2));
title 'c, F_s = 10.5'
subplot 414
plot(t2(range2), d(range2));
title 'd, F_s = 10.5'
```

从图中可以看到，我们恢复出的信号确实是原信号的按位取反。

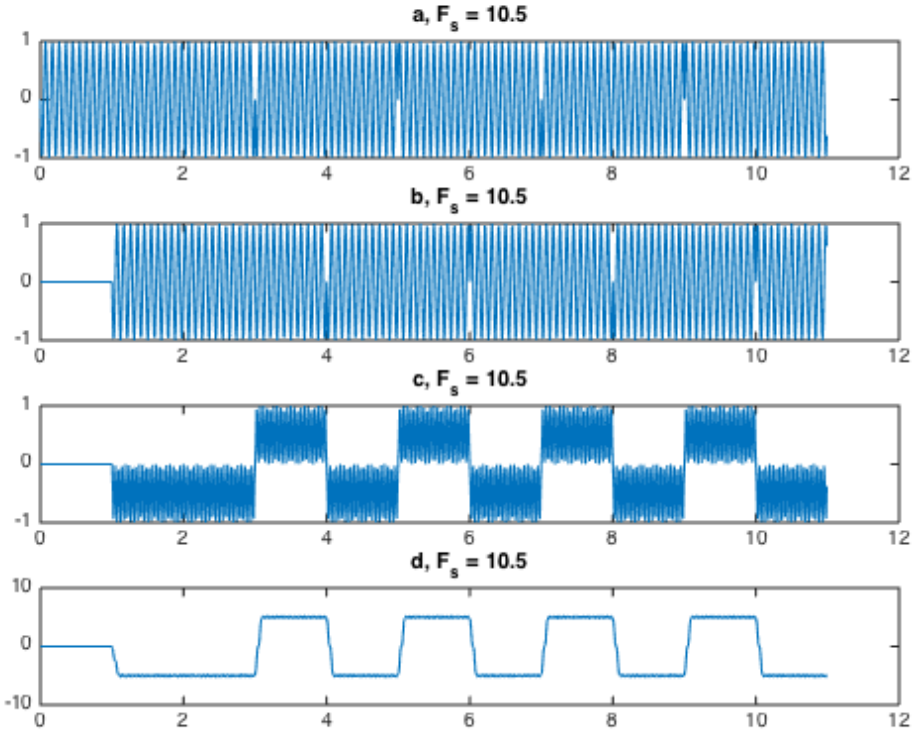

图 10: 载波频率为 10.5 倍比特率时 DPSK 波形