

IoT Communication Experiment

物联网通信实验报告

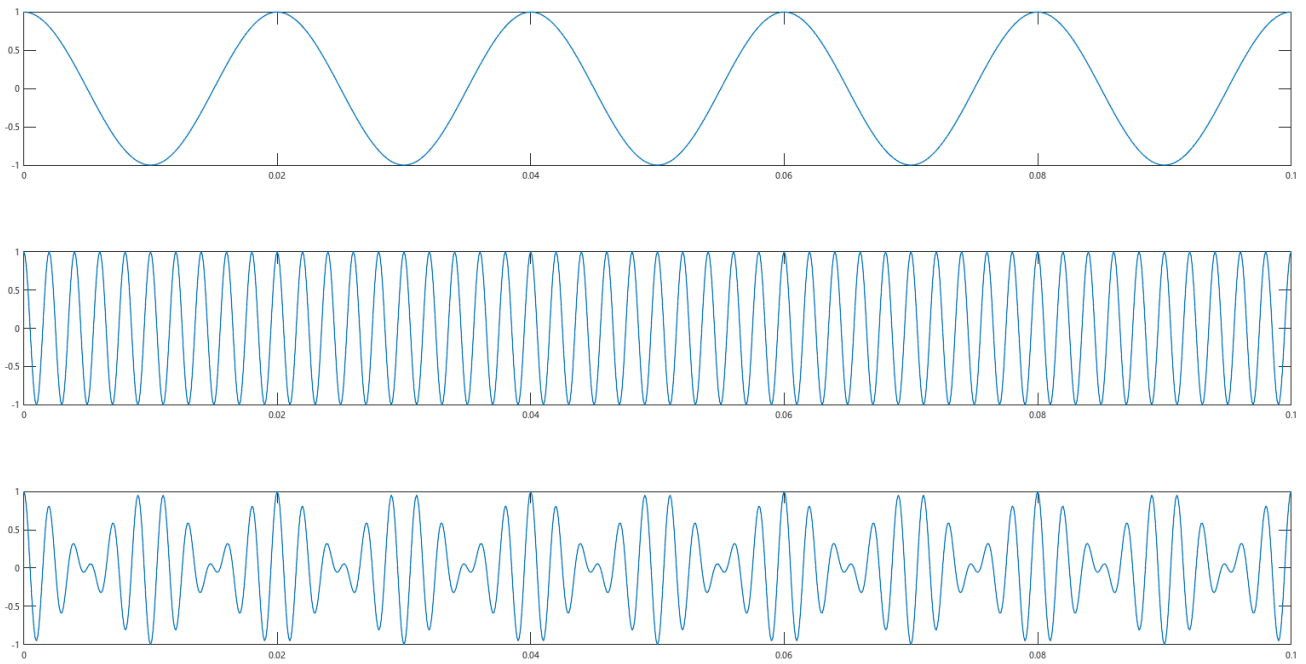
201616070320 | 物联网1603 | 郭治洪 | 指导老师: 王佳莹
使用 Octave 或 Matlab 制图 | Markdown 书写
MathJax 提供数学公式支持 | FlowChart 提供流程图支持

练习 1 画出三个图形, 排成上中下, 依次为 $\cos(100\pi t)$, $\cos(1000\pi t)$ 以及 $\cos(100\pi t) \cos(1000\pi t)$ (注: π 可直接用 pi 表示)。

Code:

```
%-----  
% IoT Communication Experiment  
% Author:GuoZhiHong  
% StudentID:201616070320  
%-----  
  
% Exercise 1  
  
t=0:0.0001:0.1; % 范围0-0.1,间隔0.0001  
x=cos(100*pi*t);  
y=cos(1000*pi*t);  
z=x.*y;  
subplot(311); % 3列1行, 第一个图  
plot(t,x);  
subplot(312); % 3列1行, 第二个图  
plot(t,y);  
subplot(313); % 3列1行, 第三个图  
plot(t,z);
```

Demo:



练习 2 画出方波及其部分傅里叶级数和的图形，方波的周期和振幅自己定义。

首先弄明白

$$T = \frac{2\pi}{\omega} = \frac{1}{f}$$

$$t = nT = n \frac{2\pi}{\omega}$$

这里我取

$$\omega = 2\pi$$

因此

$$T = 1$$

$$f = 1$$

因此方波的分段函数是

$$f(n) = \begin{cases} 1, & 0 < t < \frac{1}{2} \\ -1, & \frac{1}{2} < t < 1 \end{cases}$$

查询Wiki百科的方波的傅里叶级数和函数：

$$x_{\text{square}}(t) = \frac{4}{\pi} \sum_{k=1}^{\infty} \frac{\sin((2k-1)2\pi ft)}{(2k-1)} = \frac{4}{\pi} \left(\sin(2\pi ft) + \frac{1}{3} \sin(6\pi ft) + \frac{1}{5} \sin(10\pi ft) + \dots \right)$$

因此我的方波的傅里叶级数和函数是：

$$f_{\text{square}}(t) = \frac{4}{\pi} \sum_{n=1}^{\infty} \frac{\sin(2\pi t(2n-1))}{(2n-1)}$$

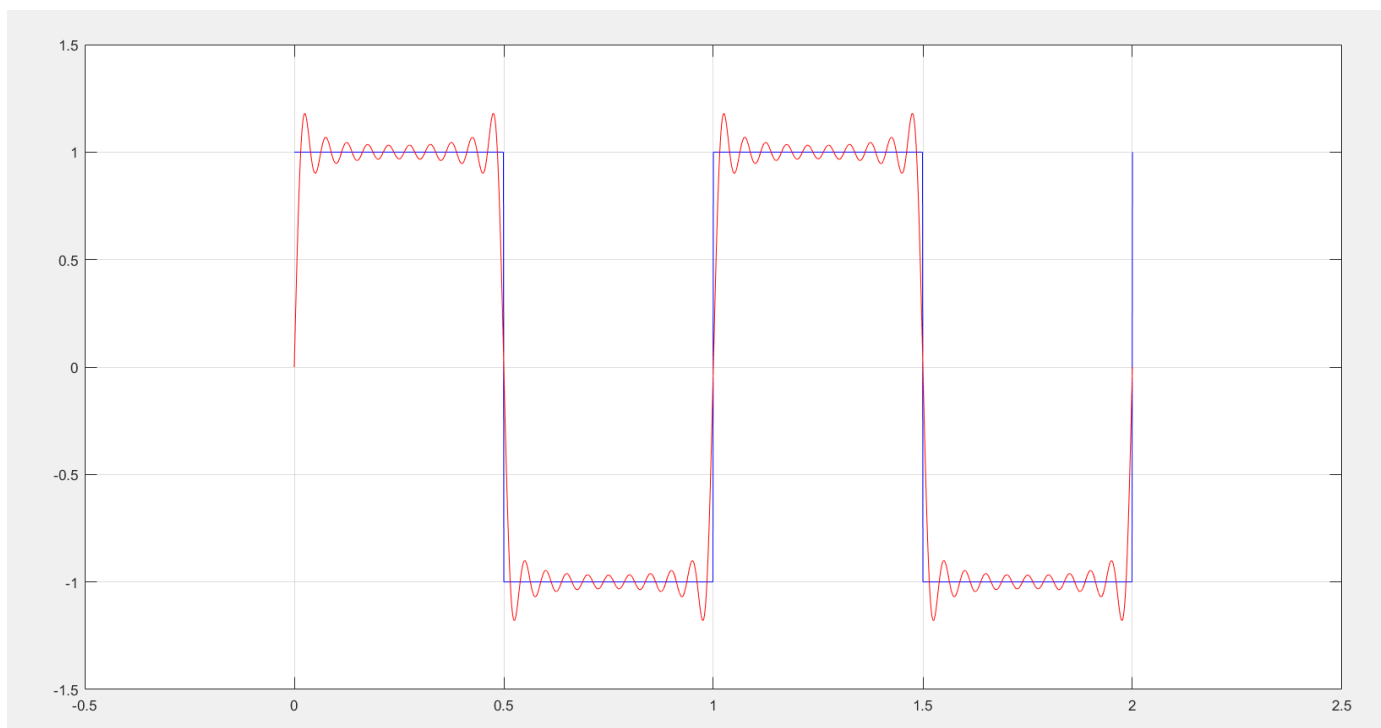
Code:

```
%-----
% IoT Communication Experiment
% Author:GuoZhiHong
% StudentID:201616070320
%-----

% Exercise 2

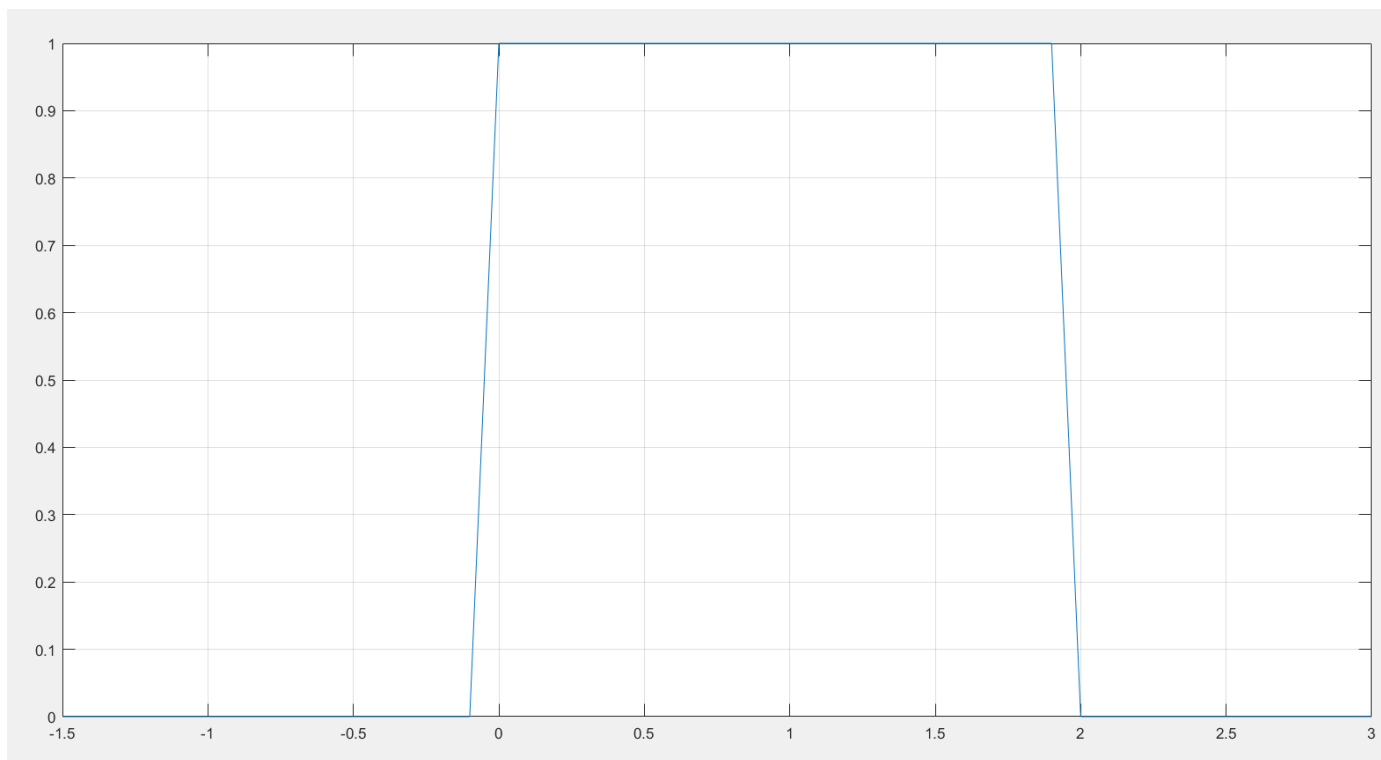
N=10;
t = 0:0.001:2;
s = square(2*pi*t);
fs = zeros(size(s));
for n = 1:N
    fs = fs + (4/pi)*((sin(2*pi*t*(2*n-1)))/(2*n-1));
end
plot(t,s,'b',t,fs,'r');
axis([-0.5,2.5,-1.5,1.5]);
grid on;
```

Demo:



练习 3 请分析时域函数 $\text{rect}((t-1)/2) \cdot \cos(20\pi t)$ 的频谱

首先做出该rect函数（又称帽子或者矩形函数）的图形



这里我的变量取值范围 $[-1.5, 3.0]$,间隔 0.1。

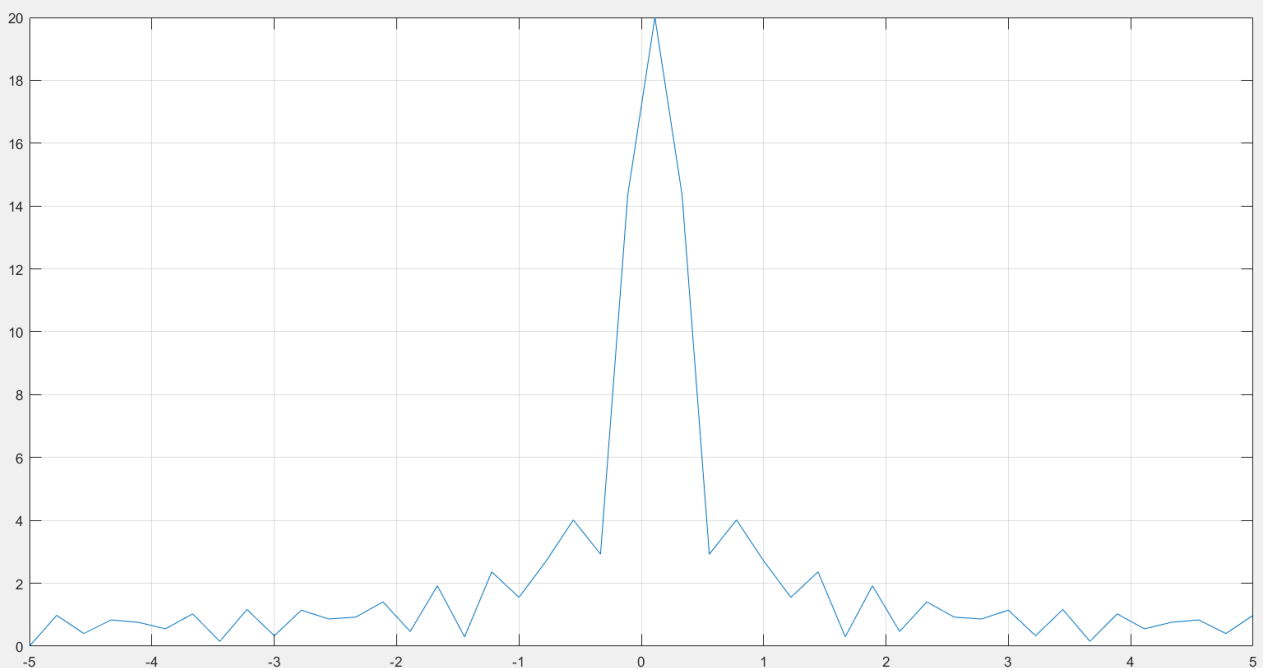
分析过程：

1. 假设用来分析的信号段有 $N = 46$ 个点,且其取样间隔为 $\Delta t = 0.1$,且取样速度为 $R = \frac{1}{\Delta t} = 10$ 。
2. 根据取样定理, 可以观察的频率范围为 $[-5, 5]$ ($[-f_0, f_0]$), $f_0 = \frac{R}{2} = 5$ 。
3. $N = 46$ 个时域点做 FFT 之后变成 $N = 46$ 个频域点, 故频率解析度, 亦即相邻频率点的间隔为 $\frac{2f_0}{N} = \frac{R}{N} = \frac{1}{(N\Delta t)} = \frac{1}{T} = \frac{5}{23}$ 。简单说, 取样时间总长的倒数就是频率解析度。
4. **要记得取样速度的一半, 是可观察的最大频率; 取样总长的倒数是频率解析度。**

Code:

```
%-----  
% IoT Communication Experiment  
% Author:GuoZhiHong  
% StudentID:201616070320  
%-----  
  
% Exercise 3  
  
t=-1.5:0.1:3.0;  
y=rectpuls((t-1)/2) .* cos(20*pi*t);  
sf = fft(y);  
sf = abs(fftshift(sf));  
f = -5:2/9:5;  
plot(f, sf);  
grid on;
```

Demo:



练习 4 试着调整不同的取样速度，例如高于或小于信号最高频率的两倍，计算还原的信号与原来信号的误差。

这里还是用老师给的例子：

以下以升余弦脉冲信号为例，来进行取样定理的简单分析。假设 $s(t)$ 为 $\alpha = 1$ 的升余弦脉冲信号，

$$s(t) = \frac{\text{sinc}(t/T) \cos(\pi t/T)}{1 - 4t^2/T^2} = \frac{\sin(2\pi t/T)}{(2\pi t/T)(1 - 4t^2/T^2)}$$

其傅里叶变换经计算后为

$$S(f) = \begin{cases} T \cos^2(\pi f T/2), & |f| < 1/T \\ 0, & |f| > 1/T \end{cases}$$

取 $T = 2$ ，则

$$s(t) = \sin(\pi t)/(\pi t(1 - t^2))$$
$$S(f) = \begin{cases} 2 \cos^2(\pi f), & |f| < 1/2 \\ 0, & |f| > 1/2 \end{cases}$$

1. 首先跟着老师，定义一个计算时域的函数，取名为 `mys.m`，一定要叫这个名字，否则无法找不到。

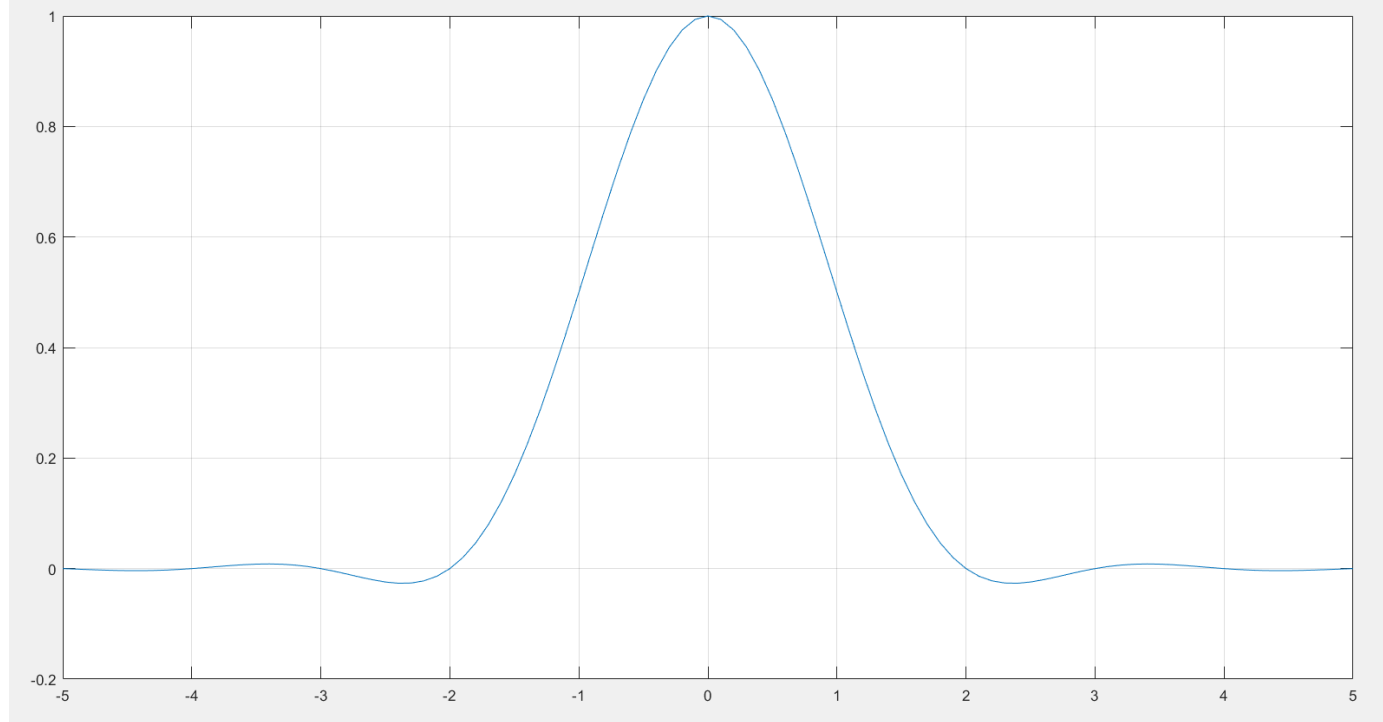
```
%-----  
% IoT Communication Experiment  
% Author:GuoZhiHong  
% StudentID:201616070320  
%-----  
  
% Exercise 4 function mys  
% This function is written by Mr.Wang  
  
function y = mys(x)  
% Calculate raise cos with a=1 and T=2  
pos1 = x== -1 | x==1; % |x|=1 须特别处理  
pos0 = x==0; % x=0 须特别处理  
x(pos0) = 99;  
x(pos1) = 99;  
y = sin(pi*x) ./ (pi*x.*(1-x.*x));  
y(pos0) = 1;  
y(pos1) = 0.5;
```

这里我注意到 $(-1, 1, 0)$ 这三个点是特殊点，计算时候把变量矩阵这三个数值改成99，计算结果很小，然后再给给定特殊值，猜测可能用来处理。

2.然后尝试调用所写的函数

```
t = -5:.1:5;  
s = mys(t);  
plot(t,s);  
grid on;
```

得到的升余弦脉冲信号



3.然后写一个计算频率的函数，取名 `mysf.m`

```
%-----  
% IoT Communication Experiment  
% Author:GuoZhiHong  
% StudentID:201616070320  
%-----  
  
% Exercise 4 function mysf  
% This function is written by Mr.Wang  
  
function y = mysf(x)  
% Calculate spectrum of raised cos with a=1, T=2  
  
pp05 = x>0.5 | x<-0.5;  
x(pp05) = 0.5;  
y = 2*cos(pi*x).*cos(pi*x);
```

4.做出他的频谱

```

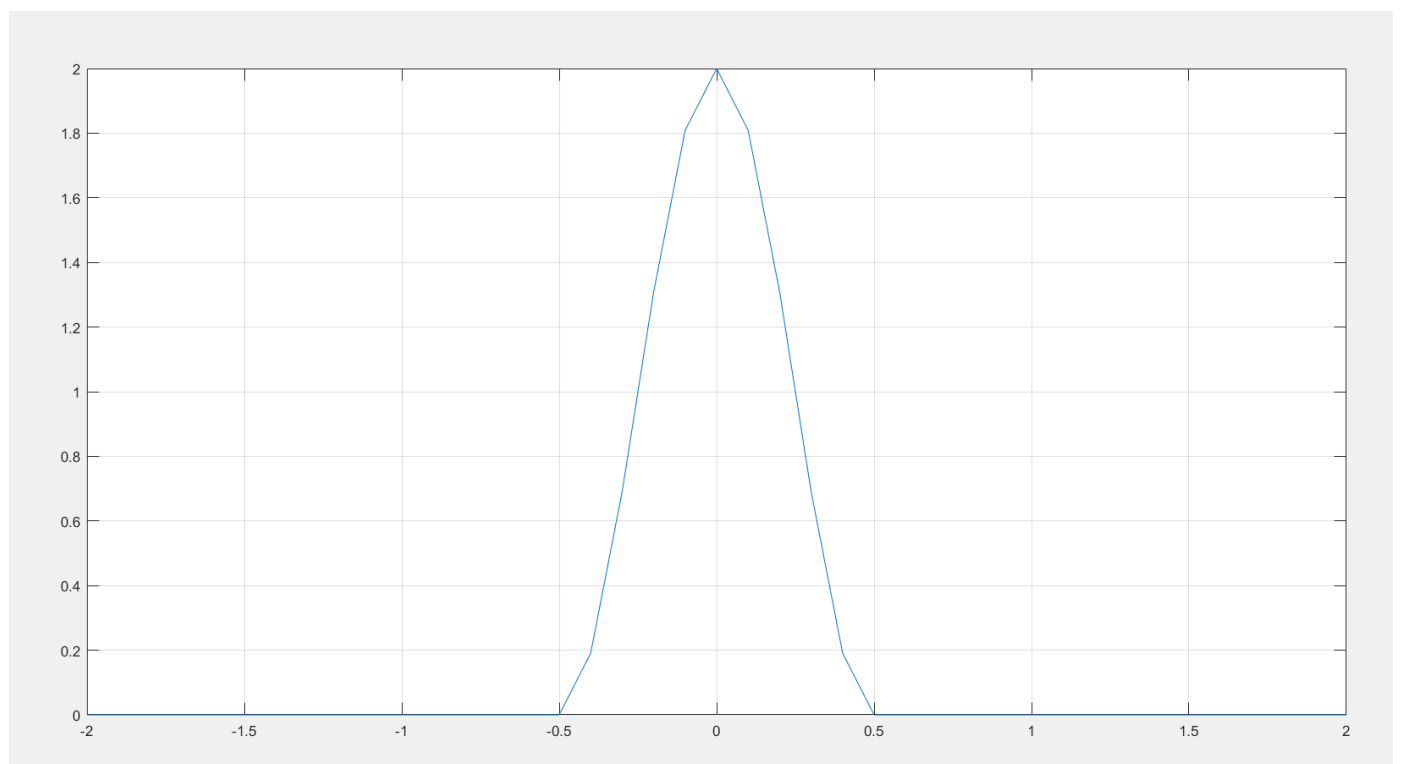
%-----
% IoT Communication Experiment
% Author:GuoZhiHong
% StudentID:201616070320
%-----

% Exercise 4

f = -2:0.1:2;
sf = mysf(f);
plot(f, sf);
grid on;

```

图形如下：



然后根据取样公式

$$X_s(f) = \sum_{k=-\infty}^{\infty} X(f - kf_s)$$

定义个计算重复频率的函数 `myssf.m`

```

%-----
% IoT Communication Experiment
% Author:GuoZhiHong
% StudentID:201616070320
%-----

```



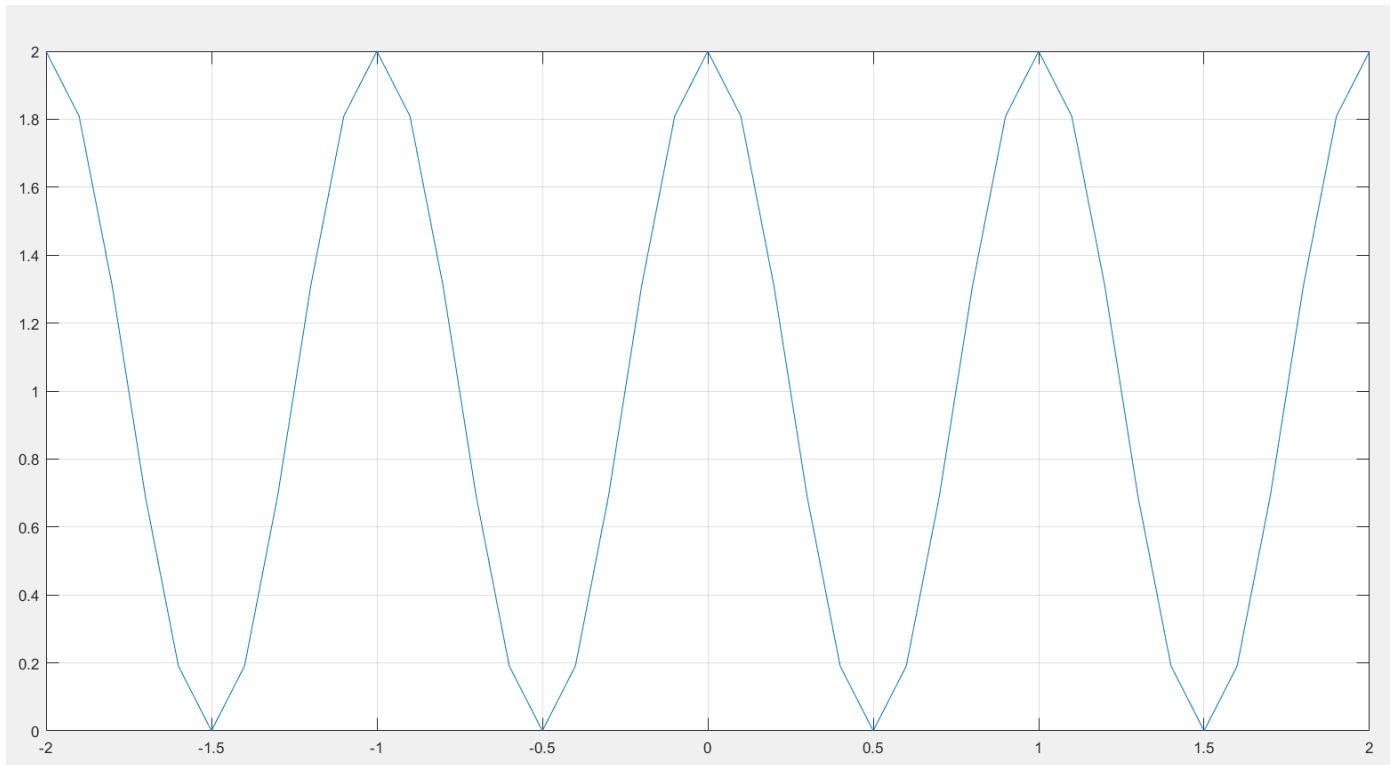
```
% Exercise 4 function mys
% This function is written by Mr.Wang

function y = mys(x)
% Calculate raise cos with a=1 and T=2
pos1 = x== -1 | x==1; % |x|=1 须特别处理
pos0 = x==0; % x=0 须特别处理
x(pos0) = 99;
x(pos1) = 99;
y = sin(pi*x) ./ (pi*x.*(1-x.*x));
y(pos0) = 1;
y(pos1) = 0.5;
```

这里可以看到老师只累加11项，取的速度 $f_s=1$ ($f_s = 1$) 然后作图。

```
f = -2:0.1:2;
sf = myssf(f,1);
plot(f, sf);
grid on;
```

这里可以观察到每 $1Hz$ 重复一次，重复4次，这可以一会解释为什么角频率重复周数 $r = 4$ ，以及角频率半周点数(π) $N = 8$ 原因。

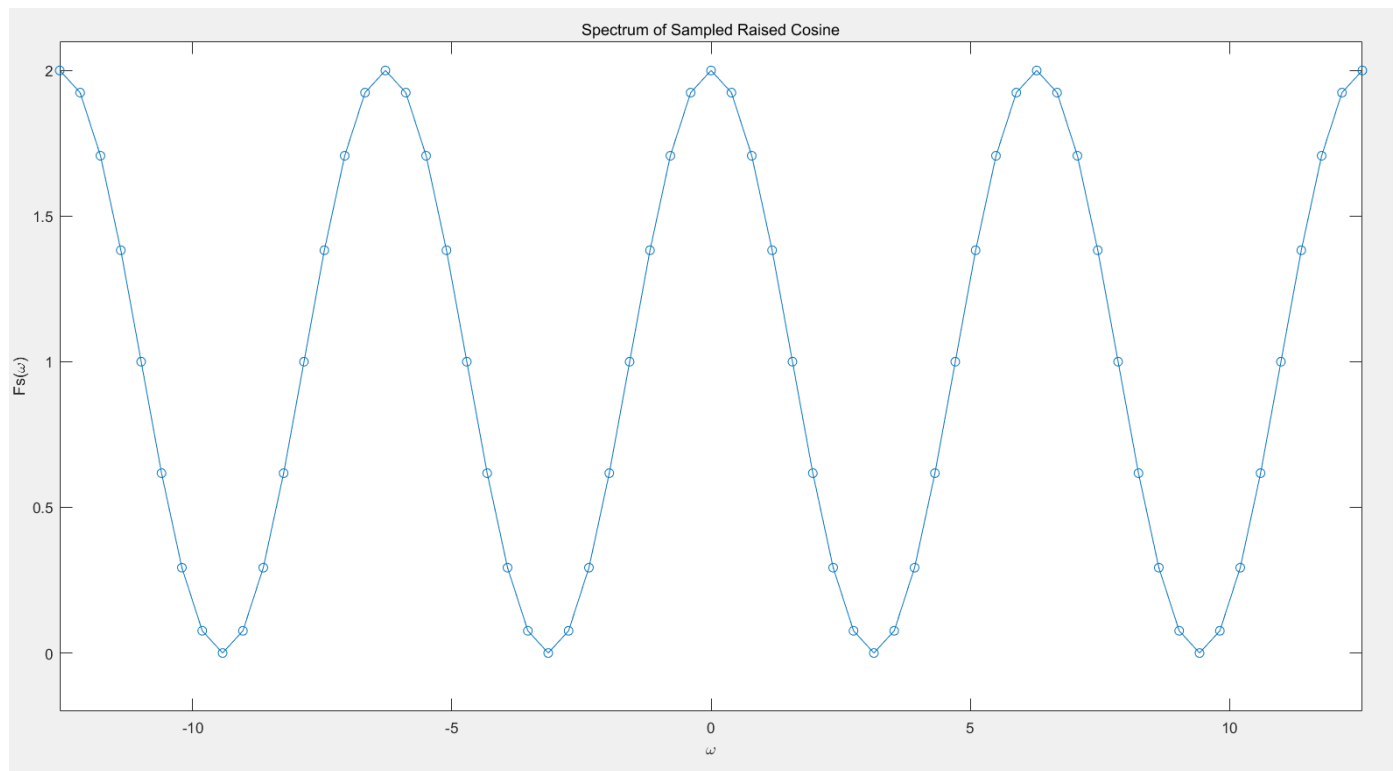


然后我们可以根据频率归一化之后在角频率 kn/N 瞬时的频谱 X_k 的大小。
公式为：

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{2\pi i}{N} kn}$$

```
Ts = 1; % 取样间隔
t2 = -4:Ts:4; % 取样时间点
fst = mys(t2); % 取样值 xn
N = 8; % 角频率半周期点数 (pi)
r = 4; % 角频率重复周数
k = -N*r:N*r; % 角频率点数范围
w = pi*k/N; % 角频率向量
Fsw = Ts*fst*exp(-j*t2'*w); % 角频率分量计算公式
plot(w, abs(Fsw), 'o-');
axis([-4*pi 4*pi -0.2 2.1]);
xlabel('\omega'), ylabel('Fs(\omega)');
title('Spectrum of Sampled Raised Cosine');
```

可以观察到我们用角频率计算的频谱与我们 `myssf.f` 使用公式累加的结果几乎相同。



如果取样够快我们可以认为时域函数和 *sinc* 函数做卷积，其公式如下：

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT) \cdot \text{sinc}\left(\frac{t - nT}{T}\right)$$

我们前面已经假设周期 $T = 2$ ，即频率 $f = 0.5$ 。根据取样定理，如果取样速度小于频率的两倍 $f_s(B) < 2f$ ，信号就会被相互干扰而无法还原。观察老师给的例子他的取样速度是频率的4倍。

因此我学着老师写法，将取样速度分别设为频率的一，二，四倍，并把他们画到一张图上。

```

%-----
% IoT Communication Experiment
% Author:GuoZhiHong
% StudentID:201616070320
%-----

% Exercise 4

S= -8:8; % 取样范围 (矩阵)
srate1 = 2; % 取样速度
srate0 = 1;
srate2 = 0.5;
st1 = S/srate1; % 取样点平均 (位置矩阵)
st0 = S/srate0; % 取样点平均 (位置矩阵)
st2 = S/srate2; % 取样点平均 (位置矩阵)
xn0 = mys(st0); % 信号函数的值 (矩阵)
xn1 = mys(st1); % 信号函数的值 (矩阵)
xn2 = mys(st2); % 信号函数的值 (矩阵)
N1 = floor(length(st1)/2); % 点数的一半, 因为累加要从负数到正数, 所以应该各一半
    % length为取样点的个数
N0 = floor(length(st0)/2); % 点数的一半, 因为累加要从负数到正数, 所以应该各一半
    % length为取样点的个数
N2 = floor(length(st2)/2); % 点数的一半, 因为累加要从负数到正数, 所以应该各一半
    % length为取样点的个数

T1 = 1/srate1; % 取样间隔应该为速度的倒数
T2 = 1/srate2; % 取样间隔应该为速度的倒数
T0 = 1/srate0; % 取样间隔应该为速度的倒数
t = -5:0.1:5; % 计算还原信号的时间点

rt0 = zeros(size(t));
for n=-N0:N0
    rt0 = rt0 + xn0((n+N0)+1)*sinc((t-n*T0)/T0);
    % N+n+1 刚好可以取矩阵xn下标从1-2*N+1的值, 刚好可以信号函数的值 (矩阵) 取完
end

rt1 = zeros(size(t));
for n=-N1:N1
    rt1 = rt1 + xn1((n+N1)+1)*sinc((t-n*T1)/T1);
    % N+n+1 刚好可以取矩阵xn下标从1-2*N+1的值, 刚好可以信号函数的值 (矩阵) 取完
end

rt2 = zeros(size(t));
for n=-N2:N2
    rt2 = rt2 + xn2((n+N2)+1)*sinc((t-n*T2)/T2);
    % N+n+1 刚好可以取矩阵xn下标从1-2*N+1的值, 刚好可以信号函数的值 (矩阵) 取完
end

plot(t,rt0,'-r.',t,rt1,'-b.',t,rt2,'-g.',t,mys(t),'k'), grid on;

```

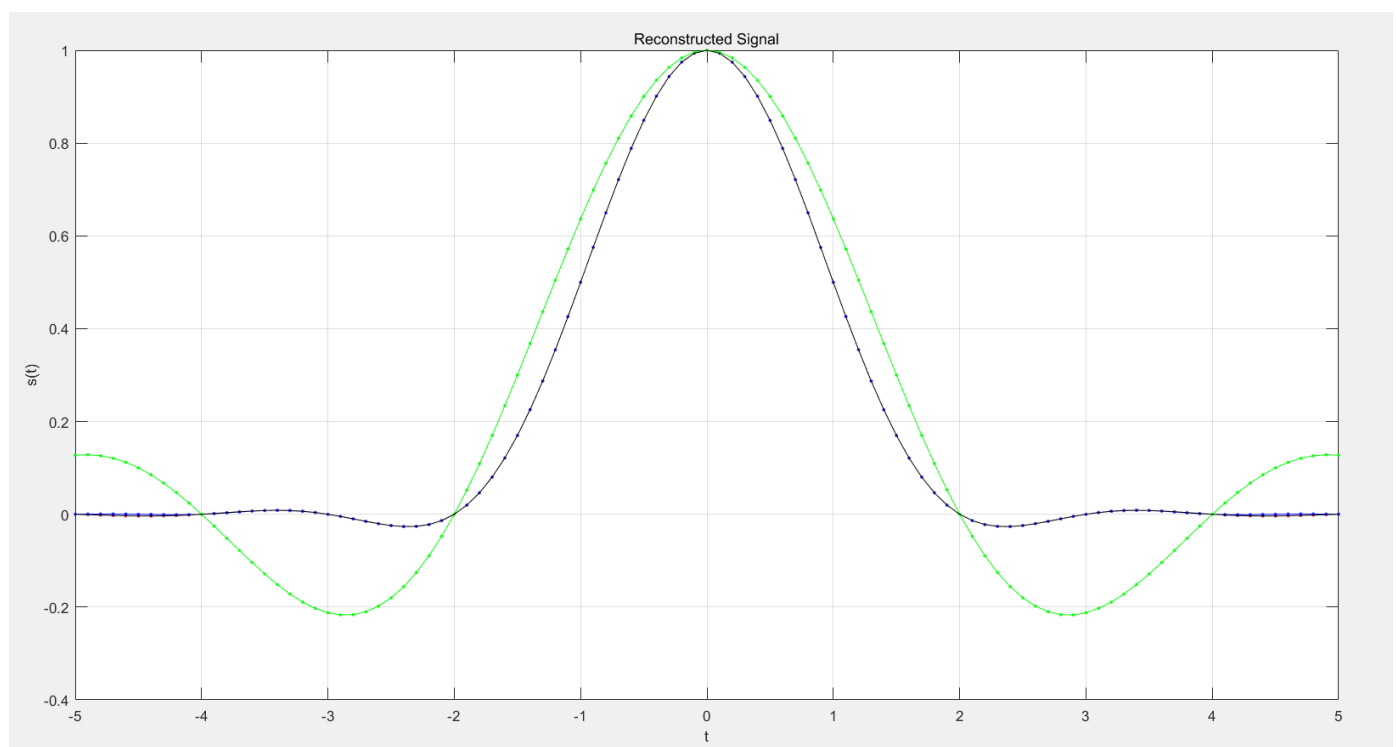
```
xlabel('t'), ylabel('s(t)');  
title('Reconstructed Signal');
```

在程序中看到 `srate1` 为4倍频率的取样速度，`srate0` 为2倍频率的取样速度，`srate` 为等于频率的取样速度，黑线为原信号函数，并且用蓝，红，绿线，黑线表示。

可以观察到当取样速度**大于**2倍频率时候误差很小，信号没有干扰，而小于2倍频率就会误差很大，信号有干扰。

至于为什么不能等于，是因为如果在最高频率上如果有冲击，等于2倍频率的情况下，如果有零界点，就可能无法恢复。

所以抽样定理的应该是采样速度**大于**最高频率的2倍，抽样之后的数字信号**完整地**保留了原始信号中的信息。



我又对抽样还原定理了解更深了一步。

最后，谢谢老师认真的教导和帮助，没有他我就完不成这个实验，老师您辛苦了。

5.假设 $g(x) = x^5 + 1$ ，信息长度为5，请把所有可能的 CRC 编码的结果表列出来，并使用侦错程式码检查一个收到的封包内容是否正确。

如果

$$c = (c_{n-1}, c_{n-2}, \dots, c_0)$$

是码字，将 c 的每个位元向左循环移位，得到

$$c^{(1)} = (c_{n-2}, c_{n-3}, \dots, c_0, c_{n-1})$$

用多项式 GF(2) 写得话

$$x \cdot c(x) \bmod (x^n + 1)$$

我们称最低次（次数为 r ）的码多项式 $g(x)$ 是循环码的生成多项式，其他的码多项式 $xg(x), x^2g(x), \dots, x^{n-r-1}g(x)$ 代表向左循环 $1, 2, \dots, n-r-1$ 个位置。连同 $g(x)$ 一共有 $n-r=k$ 个多项式，他们线性无关。

假设使用生成多项式 $g(x)$ ，其的次数为 $r = n - k$ ，假设使用的封包信息 m 有 k 个位元，对应多项式 $m(x)$ 。

首先 $m(x)$ 乘上 x^r （左移 r 个位元），得到 $p(x) = x^r m(x)$ ，用 $p(x)$ 除以 $g(x)$ 的余式 $t(x)$ ，则 $p(x) - t(x)$ 除以 $g(x)$ 余式为 0，也就是 $p(x) - t(x)$ 应该就是 $g(x)$ 的倍式。因为在 GF(2) 的场域，加法和减法无异，因此把 $p(x)$ 调整为 $p(x) + t(x)$ ，就可以得到码多项式。由于 $g(x)$ 是 r 次多项式，所以 $t(x)$ 的最高次数是 $r - 1$ 。以下是运算规则：

1. 把信息多项式 $m(x)$ 往左移 r 个位置，得到的多项式为 $p(x) = x^r m(x)$ 。
2. 把 $p(x)$ 除以 $g(x)$ 得到余式 $t(x)$ ，其对应的 r 位元二元向量为 t 。
3. 把这 r 个位元向量 t 补在 m 的后面，实际上就是一个码字了。

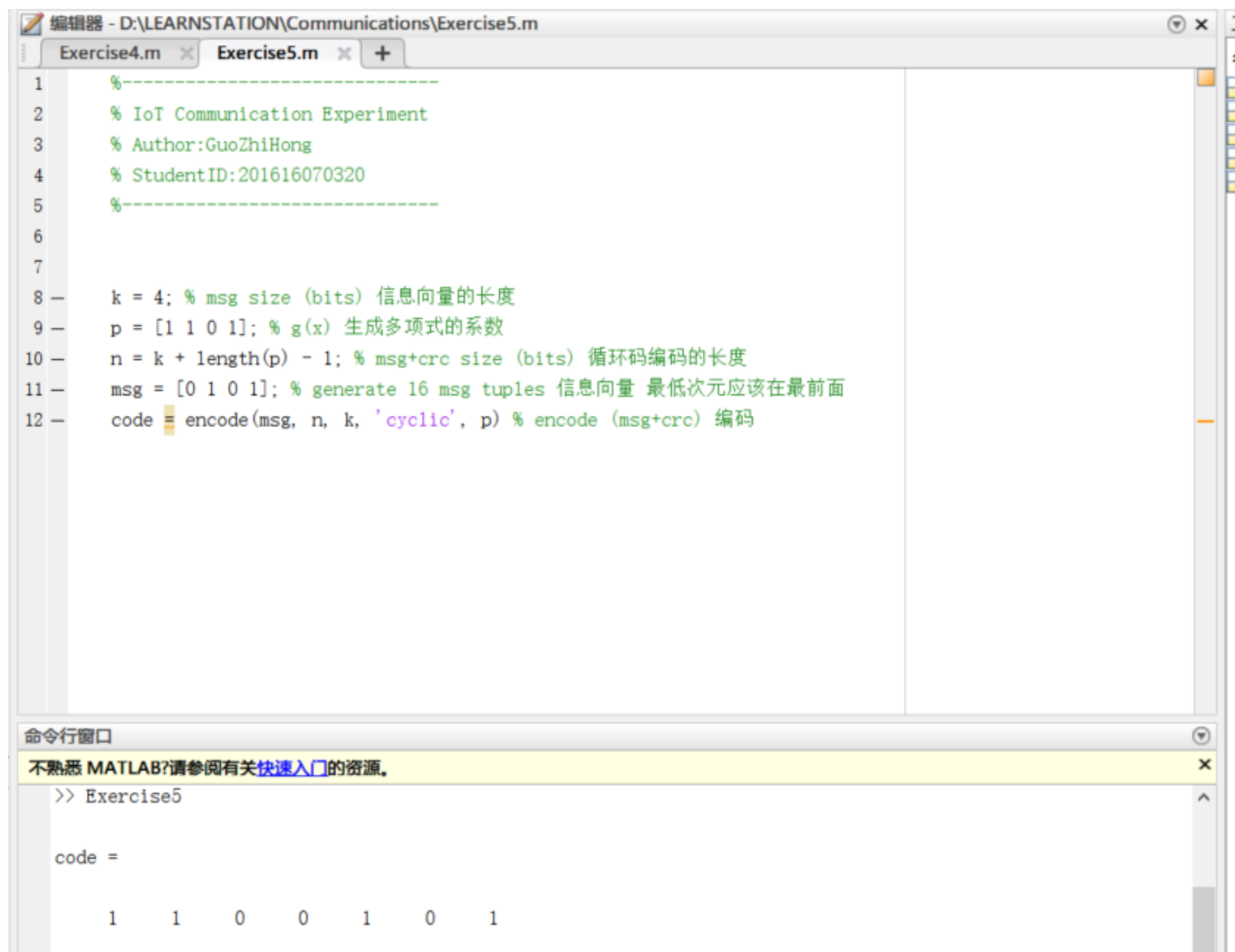
MATLAB里面有 `encode` 函数，可以帮助实现，用法如下：

```
encode(m,n,k,'cyclic',p);
% m 是信息向量
% n = k + r 是码长
% k 是信息长度
% p 是生成多项式的系数
```

假设信息向量为 1010，生成多项式为 $x^3 + x + 1$ ， $k = 4$ ， $n = 7$ ，则得到的 `code = 101001` 就是编码后的结果：

```
%-----
% IoT Communication Experiment
% Author:GuoZhiHong
% StudentID:201616070320
%-----

k = 4; % msg size (bits) 信息向量的长度
p = [1 1 0 1]; % g(x) 生成多项式的系数
n = k + length(p) - 1; % msg+crc size (bits) 循环码编码的长度
msg = [0 1 0 1]; % generate 16 msg tuples 信息向量 最低次元应该在最前面
code = encode(msg, n, k, 'cyclic', p) % encode (msg+crc) 编码 最低次元应该在最前面
```



The screenshot shows the MATLAB Editor window with a file named 'Exercise5.m'. The code defines a CRC encoding function. It sets `k = 4` for message size, `p = [1 1 0 1]` for the generator polynomial, and `n = k + length(p) - 1` for the total length. The message `msg = [0 1 0 1]` is defined. The `code` variable is assigned the result of `encode(msg, n, k, 'cyclic', p)`. The Command Window shows the execution of `>> Exercise5`, resulting in `code =` followed by the vector `[1 1 0 0 1 0 1]`.

```
1 %-----
2 % IoT Communication Experiment
3 % Author:GuoZhiHong
4 % StudentID:201616070320
5 %-----
6
7
8 k = 4; % msg size (bits) 信息向量的长度
9 p = [1 1 0 1]; % g(x) 生成多项式的系数
10 n = k + length(p) - 1; % msg+crc size (bits) 循环码编码的长度
11 msg = [0 1 0 1]; % generate 16 msg tuples 信息向量 最低次元应该在最前面
12 code = encode(msg, n, k, 'cyclic', p) % encode (msg+crc) 编码
```

命令行窗口

不熟悉 MATLAB? 请参阅有关[快速入门](#)的资源。

```
>> Exercise5

code =

    1    1    0    0    1    0    1
```

注意：最低次元应该出现到最前面。即输入的信息向量和输出的编码结果是最低次元应该在最前面。

这里可以使用 `fliplr(x)` 完成向量镜像左右反转。

```
%-----
% IoT Communication Experiment
% Author:GuoZhiHong
% StudentID:201616070320
%-----

k = 4; % msg size (bits) 信息向量的长度
p = [1 1 0 1]; % g(x) 生成多项式的系数
n = k + length(p) - 1; % msg+crc size (bits) 循环码编码的长度
msg = [1 0 1 0]; % generate 16 msg tuples 信息向量 最低次元应该在最前面
msg = fliplr(msg);
code = encode(msg, n, k, 'cyclic', p); % encode (msg+crc) 编码
% 但是注意要编码的信息向量必须最低次元要在最前面，编码在之后结果也是最低次元在最前面
code = fliplr(code)
```

```

%-----
% IoT Communication Experiment
% Author:GuoZhiHong
% StudentID:201616070320
%-----

k = 4; % msg size (bits) 信息向量的长度
p = [1 1 0 1]; % g(x) 生成多项式的系数
n = k + length(p) - 1; % msg+crc size (bits) 循环码编码的长度
msg = [1 0 1 0]; % generate 16 msg tuples 信息向量 最低次元应该在最前面
msg = fliplr(msg);
code = encode(msg, n, k, 'cyclic', p); % encode (msg+crc) 编码
code = fliplr(code)

```

```

>> Exercise5

code =

     1     0     1     0     0     1     1

```

注意：注意要编码的信息向量必须最低次元要在最前面，编码在之后结果也是最低次元在最前面。

然后学老师把 0000, 0001, ..., 1111 等16种信息向量都打印出来，[这里为了看着方便我镜像翻转了向量](#)。

```

%-----
% IoT Communication Experiment
% Author:GuoZhiHong
% StudentID:201616070320
%-----

% Show CRC coding Table
k = 4; % msg size (bits) 信息向量的长度
p = [1 1 0 1]; % g(x) 生成多项式的系数
n = k + length(p) - 1; % msg+crc size (bits) 循环码编码的长度
smsg = de2bi(0:15) % generate 16 msg tuples 十进制转二进制向量
code = encode(smsg, n, k, 'cyclic', p); % encode (msg+crc) 编码
code = fliplr(code) % 向量镜像反转输出结果

```

结果如下：

```
msg =
```

```
0 0 0 0
1 0 0 0
0 1 0 0
1 1 0 0
0 0 1 0
1 0 1 0
0 1 1 0
1 1 1 0
0 0 0 1
1 0 0 1
0 1 0 1
1 1 0 1
0 0 1 1
1 0 1 1
0 1 1 1
1 1 1 1
```

```
code =
```

```
0 0 0 0 0 0 0
0 0 0 1 0 1 1
0 0 1 0 1 1 0
0 0 1 1 1 0 1
0 1 0 0 1 1 1
0 1 0 1 1 0 0
0 1 1 0 0 0 1
0 1 1 1 0 1 0
1 0 0 0 1 0 1
1 0 0 1 1 1 0
1 0 1 0 0 1 1
1 0 1 1 0 0 0
1 1 0 0 0 1 0
1 1 0 1 0 0 1
1 1 1 0 1 0 0
1 1 1 1 1 1 1
```

我们纠错就是通过信息内容重新编码然后和收到信息比较，如果不一样就是有错，要求重新传输。

下面讨论这道题：

先把正确结果打印出来

Code:


```

%-----
% IoT Communication Experiment
% Author:GuoZhiHong
% StudentID:201616070320
%-----

% Show CRC coding Table
k = 5; % msg size (bits) 信息向量的长度
p = [1 0 0 0 1]; % g(x) 生成多项式的系数
n = k + length(p) - 1; % msg+crc size (bits) 循环码编码的长度
smsg = de2bi(0:15) % generate 16 msg tuples 十进制转二进制向量
code = encode(smsg, n, k, 'cyclic', p); % encode (msg+crc) 编码
code = fliplr(code) % 向量镜像反转输出结果

```

Demo:

```
>> Exercise5
```

```
smsg =
```

0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
1	1	0	0	0
0	0	1	0	0
1	0	1	0	0
0	1	1	0	0
1	1	1	0	0
0	0	0	1	0
1	0	0	1	0
0	1	0	1	0
1	1	0	1	0
0	0	1	1	0
1	0	1	1	0
0	1	1	1	0
1	1	1	1	0
0	0	0	0	1
1	0	0	0	1
0	1	0	0	1
1	1	0	0	1
0	0	1	0	1
1	0	1	0	1
0	1	1	0	1
1	1	1	0	1
0	0	0	1	1
1	0	0	1	1
0	1	0	1	1
1	1	0	1	1

```

0    0    1    1    1
1    0    1    1    1
0    1    1    1    1
1    1    1    1    1

```

```
code =
```

```

0    0    0    0    0    0    0    0    0    0
0    0    0    0    1    0    0    0    0    1
0    0    0    1    0    0    0    0    1    0
0    0    0    1    1    0    0    0    1    1
0    0    1    0    0    0    0    1    0    0
0    0    1    0    1    0    0    1    0    1
0    0    1    1    0    0    0    1    1    0
0    0    1    1    1    0    0    1    1    1
0    1    0    0    0    0    1    0    0    0
0    1    0    0    1    0    1    0    0    1
0    1    0    1    0    0    1    0    1    0
0    1    0    1    1    0    1    0    1    1
0    1    1    0    0    0    1    1    0    0
0    1    1    0    1    0    1    1    0    1
0    1    1    1    0    0    1    1    1    0
0    1    1    1    1    0    1    1    1    1
1    0    0    0    0    1    0    0    0    0
1    0    0    0    1    1    0    0    0    1
1    0    0    1    0    1    0    0    1    0
1    0    0    1    1    1    0    0    1    1
1    0    1    0    0    1    0    1    0    0
1    0    1    0    1    1    0    1    0    1
1    0    1    1    0    1    0    1    1    0
1    0    1    1    1    1    0    1    1    1
1    1    0    0    0    1    1    0    0    0
1    1    0    0    1    1    1    0    0    1
1    1    0    1    0    1    1    0    1    0
1    1    0    1    1    1    1    0    1    1
1    1    1    0    0    1    1    1    0    0
1    1    1    0    1    1    1    1    0    1
1    1    1    1    0    1    1    1    1    0
1    1    1    1    1    1    1    1    1    1

```

然后进行测试

Code:

```

%-----
% IoT Communication Experiment
% Author:GuoZhiHong

```

```

% StudentID:201616070320
%-----

% 注意在新版MATLAB 字符串使用单引号 str='string';

k = 5; % msg size (bits) 信息向量的长度
p = [1 0 0 0 0 1]; % g(x) 生成多项式的系数
n = k + length(p) - 1; % msg+crc size (bits) 循环码编码的长度
while true
    m = input('Received bits: ', 's'); % Input received bits 输入信息复制给m变量
    % 正常情况下 信息码在前面 校验码在后面
    m = fliplr(m) - '0'; % convert to a binary vector 转换成位二进制向量 这里进行向量镜像翻转
    if length(m) ~= n % input check 检测输入是否满足, 防止出现索引错误
        disp('Input Error!')
    else
        msg = m((n-k+1):n); % get msg bits 因为二进制向量翻转后校验码在前面 信息码在后面
        code = encode(msg, n, k, 'cyclic', p); % re-encode 重新编码
        if m == code % check if it is the same 检测是否错误
            disp('Correct!') % correct
        else
            disp('Error!') % error
        end
    end
end
end

```

Demo:

```

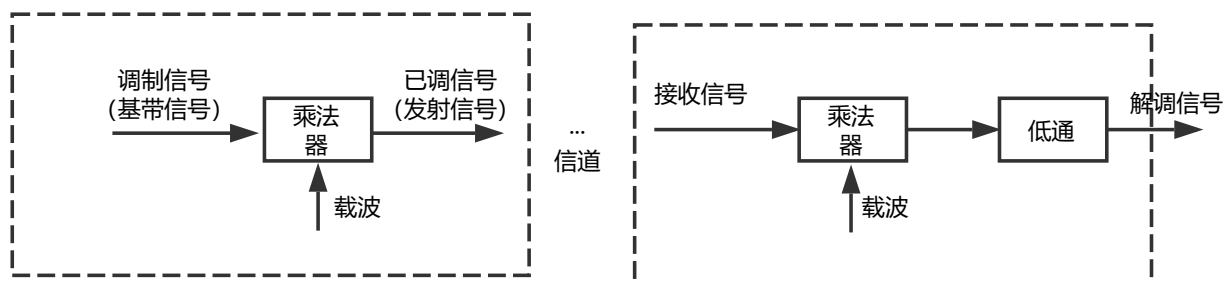
>> Exercise5
Received bits:
Input Error!
Received bits: 1
Input Error!
Received bits: 0000000000
Correct!
Received bits: 1111111111
Correct!
Received bits: 1111111110
Error!
Received bits: 1111011110
Correct!
Received bits: 1101111011
Correct!
Received bits:

```

6.假设 $m(t)$ 是抽样定理一节所述的升余弦函数，请设定适当的参数值，重做上述的实验，并画出实验结果。

首先我通过查阅资料和研究老师给的实验指导书重新复习了调制解调的原理。

简单的调制解调原理图



从原理图可以观察到我们需要的调制信号会和载波相乘变成已经调制好的发射信号经过信道传输，在接收端同样再和载波相乘然后通过一个低通滤波器，过滤掉高频部分，然后就可以得到解调信号。如果想要取得更好的调制效果我们还可可在调制发射端使用带通进行信号过滤，防止噪声，数字信号在接收端使用抽样判决器还原等。

而根据所乘的载波不同，我们可以分为**幅度**，**频率**，**相位**调制。

而我们这次实验分析的为模拟信号的幅度解调调制，使用的是相干解调法。除此之外，幅度调制还有包络调制的方式。

在老师给的实验指导书的例子上，他用 $m(t)$ 表示要解调的波形，这就是基带信号。

$$m(t) = A_m \cos(2\pi f_m t)$$

他使用一个高幅度的弦波

$$c(t) = A_c \cos(2\pi f_c t)$$

作为载波，然后相乘并进行和差化积观察到调制的信号有三个部分：载波 $c(t)$ ，频率略高略低于载波的边带弦波。这些**边带可以在相干调制中再和载波相乘后用低通滤波器过滤掉，只留下载波 $c(t)$ 部分。**

他实际给的例子这个参数中，在调制过程中，他已经先改变了调制信号（基带信号的）振幅，然后又乘上了一个高幅度的载波，经过 AWGN 信道，这个实际是标准的白噪声信道。然后他又乘了载波并通过低通滤波器，得到解调信号。注意这里解调信号实际并不是原始调制信号（基带信号），因为他对调制信号也做了一些处理，如果要得到真正的原始调制信号（基带信号）在滤波器后面还要做运算。

这里我们只是了解原理，并不需要特别深入，因此老师并没有介绍。我就大致了解一下这些部分，知道即可。

回到要做的实验中来：

假设 $s(t)$ 为 $\alpha = 1, T = 2$ 升余弦脉冲信号
表达式为：

$$s(t) = \frac{\sin(\pi t)}{\pi t(1-t^2)}, S(f) = \begin{cases} 2\cos^2(\pi f), |f| < \frac{1}{2} \\ 0, |f| > \frac{1}{2} \end{cases}$$

在上面实验中，我注意到 t 有些范围没法取，于是我选取其他范围避开。
同样的我使用高幅度弦波作为载波。

Code:

```
%-----  
% IoT Communication Experiment  
% Author:GuoZhiHong  
% StudentID:201616070320  
%-----  
  
t=2:0.001:3; % 取样时间  
SNR=0; % 信噪比 (dB)  
Am=0.5; % 信号幅度  
Ac=10; % 载波幅度  
fc=5; % 载波频率  
  
message=Am*sin(pi*t)/(pi*t.*(1-t.*t));  
carrier=Ac*cos(2*pi*fc.*t);  
AM_signal=[1+message].*carrier;  
% 假设经过 AWGN 信道  
RX_signal=awgn(AM_signal, SNR); % awgn 函数可模拟 AWGN 信道  
  
% 相干解调  
R2_signal = RX_signal.*carrier; % 再乘一次载波  
wn = .02; % 滤波器参数  
[b,a] = butter(2,wn); % 截止频率 = 50*0.02 = 1  
demod_signal = filter(b,a,R2_signal); % 通过滤波器  
  
subplot(311)  
plot(t,AM_signal)  
grid off  
title('AM 信号')  
xlabel('时间'), ylabel('幅度')  
  
subplot(312)
```

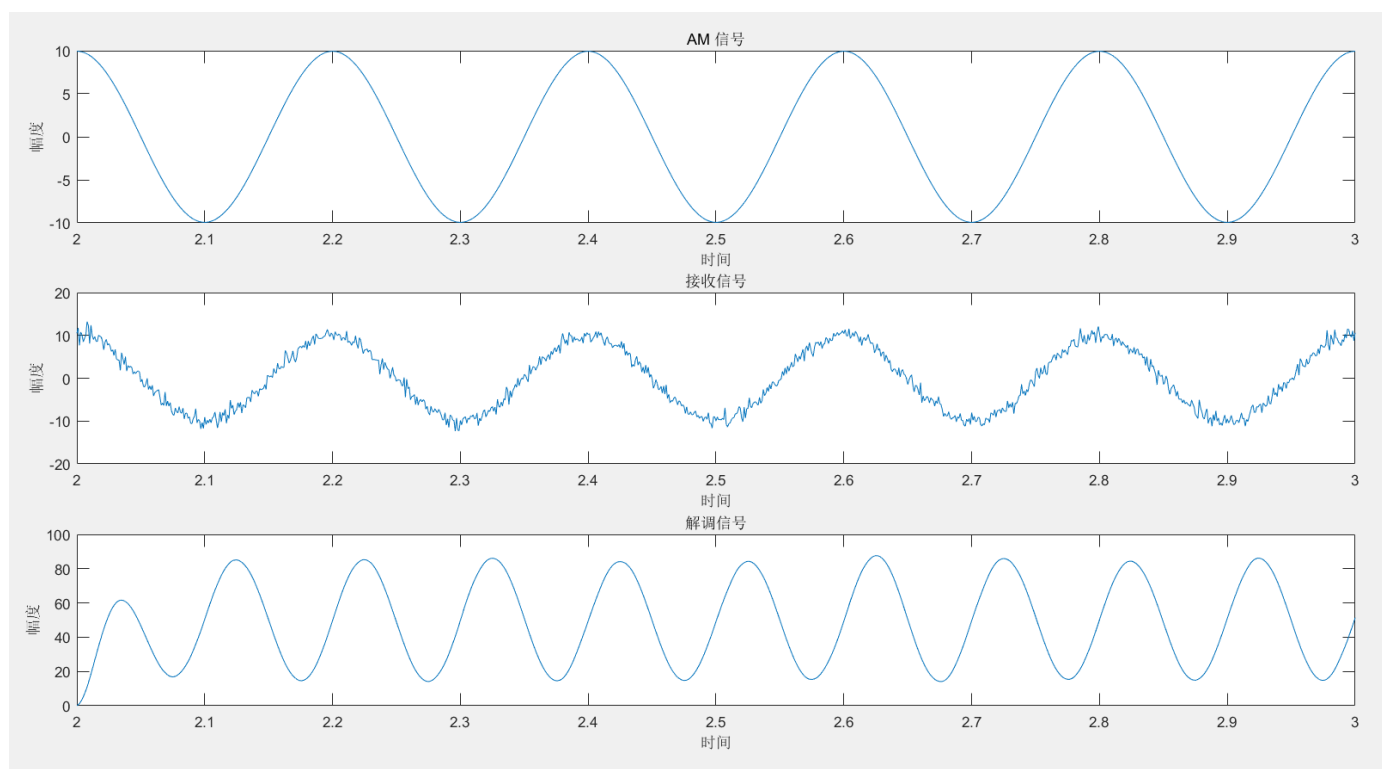
```

plot(t,RX_signal)
grid off
title('接收信号')
xlabel('时间'), ylabel('幅度')

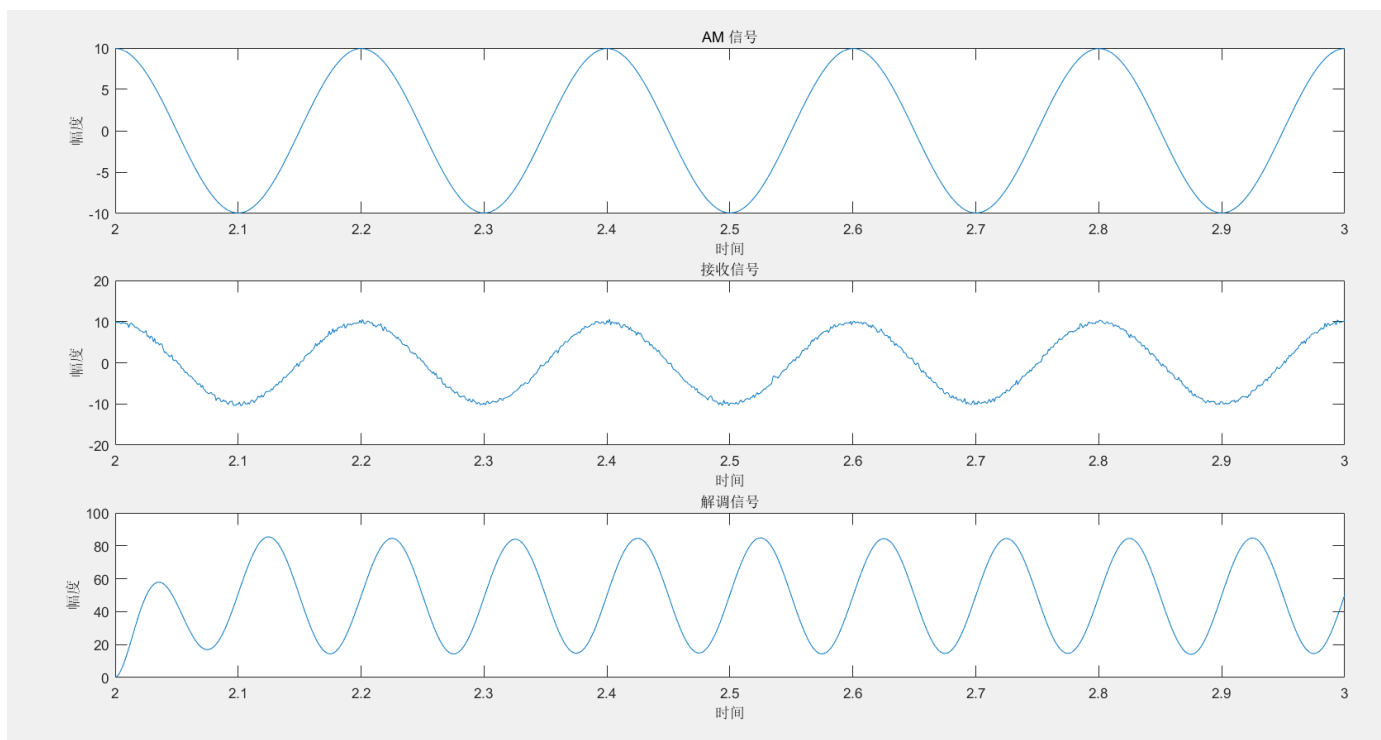
subplot(313)
plot(t,demod_signal)
grid off
title('解调信号')
xlabel('时间'), ylabel('幅度')

```

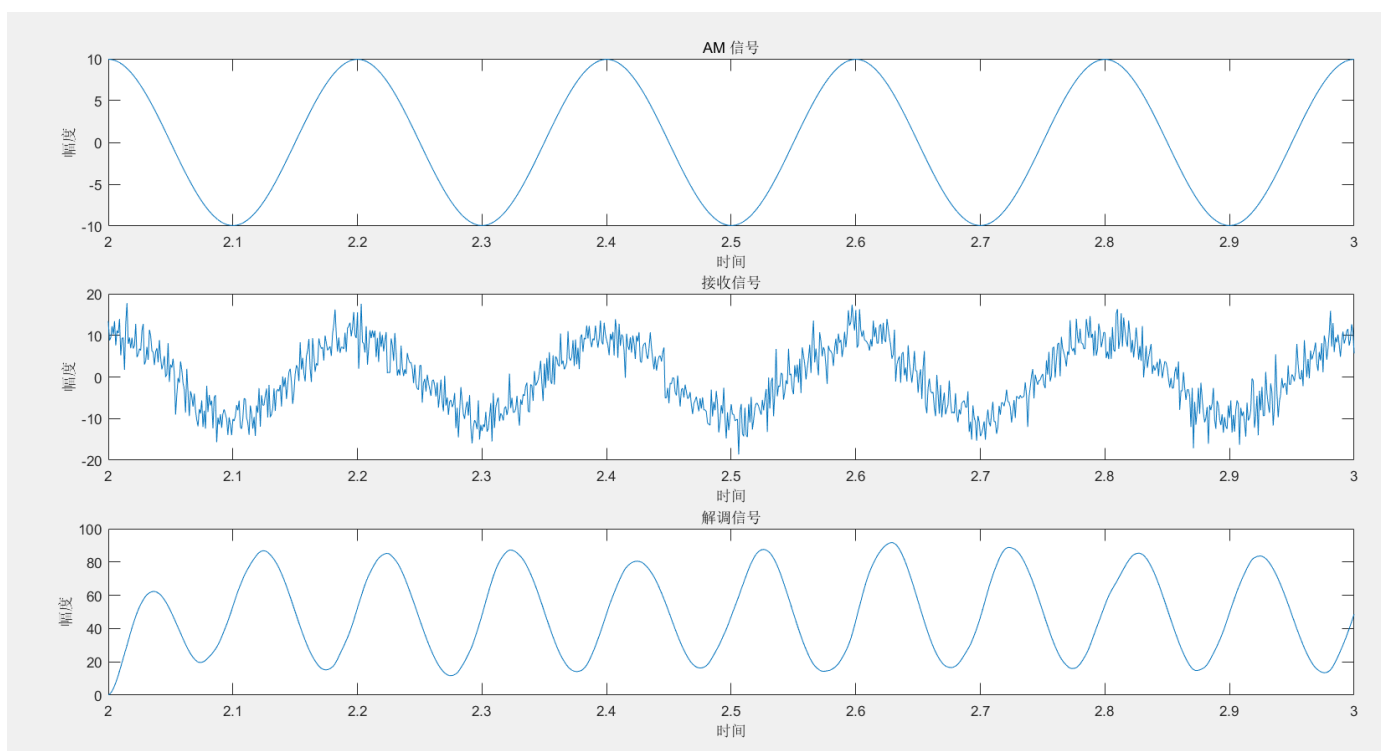
Demo:



SNR = 0dB



SNR = 10dB



SNR = -10dB

- END -
Thanks for Mr.Wang teaching and help.
2018.5