北京航空航天大學
BEIHANG UNIVERSITY

通信原理
MATLAB 软件作业一

抑制载波双边带幅度调制与
相干解调系统的仿真

学　　院　　电子信息工程学院
作　　者　　何沃洲
学　　号　　13021264

2016 年 4 月

目　录

（一）实验原理

1.1 抑制载波双边带幅度调制（DSB-SC）的原理

调制是指将发送的信号附加在高频振荡的载波上，以便于由天线发射。其中，振幅调制（AM）是一种线性调制，由调制信号去控制载波的振幅，使其随调制信号做线性变化。为了提高传输的效率，不附加直流分量，便得到了载波受到抑制的双边带调幅波（DSB-SC）。

设正弦载波为

$$c(t) = A\cos(\omega_c t + \varphi_0) \tag{1}$$

式中，A 为载波幅度；ω_c 为载波角频率；φ_0 为载波初始相位(通常令 φ_0 =0)。基带调制信号为 $m(t)$。根据调制的定义，振幅调制信号（已调信号）一般可以表示为

$$s_m(t) = Am(t)\cos(\omega_c t) \tag{2}$$

设调制信号 $m(t)$ 的频谱为 $M(\omega)$，则已调信号 $s_m(t)$ 的频谱 $S_m(\omega)$ 可表示为：

$$S_m(\omega) = \frac{A}{2}[M(\omega+\omega_c) + M(\omega-\omega_c)] \tag{3}$$

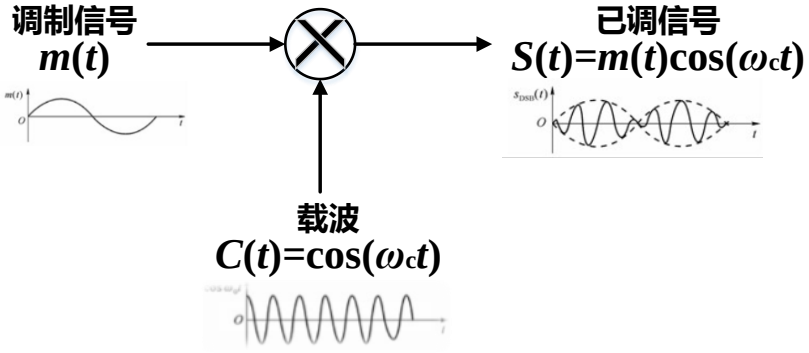

图 1　DSB 的原理示意图

1.2 DSB-SC 的相干解调原理

相干解调是指利用乘法器，输入一路与载频相干（同频同相）的参考信号与载频相乘。相干解调的一般模型如图 2 所示。

设输入为 DSB-SC 信号

$$S_m(t) = S_{DSB}(t) = m(t)\cos(\omega_c t + \varphi_0) \tag{4}$$

乘法器输出为

$$\begin{aligned}\rho(t) &= S_{DSB}(t) = m(t)\cos(\omega_c t + \varphi_0)\cos(\omega_c t + \varphi) \\ &= \frac{1}{2}m(t)[\cos(\varphi - \varphi_0) + \cos(2\omega_c t + \varphi_0 + \varphi)]\end{aligned} \tag{5}$$

通过低通滤波器后

$$m_0(t) = \frac{1}{2}m(t)\cos(\varphi_0 - \varphi) \tag{6}$$

当$\varphi_0 = \varphi =$常数时，解调输出信号为

$$m_0(t) = \frac{1}{2}m(t) \tag{7}$$

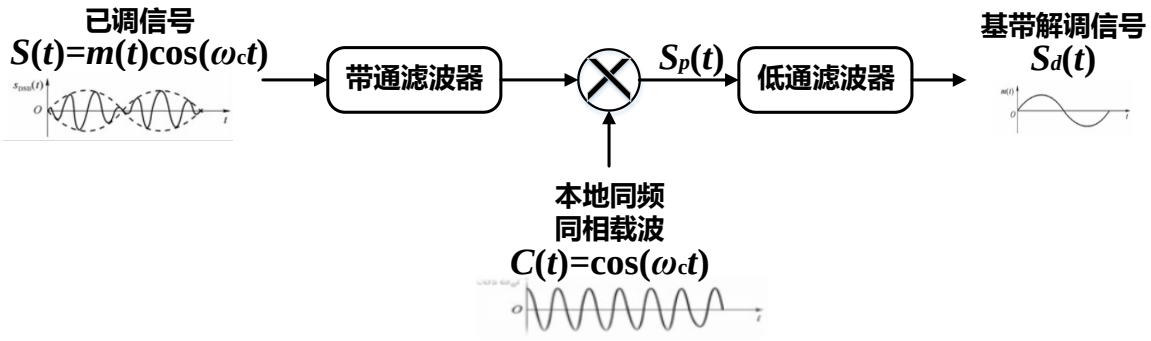

图 2　相干解调的原理示意图

（二）MATLAB 程序及仿真结果

2.1 变量初始化

```
%--------------------------------------------------------------------------
% 仿真参数设置
%--------------------------------------------------------------------------
clear all;
fm=3e3;   %调制单音信号的频率
Am=1;     %调制单音信号的幅度
fc=30*fm; %载波频率
fs=10*fc; %采样频率
ts=1/fs;  %采样的时间步长
t0=10e-4; %仿真时长
```

```
L=t0/ts;  %数据长度
t=ts:ts:t0;% 时间轴序列
N=2^16;   %FFT点数
N1_bpf=6150; %设置解调器前端带通滤波器的通带下边界
N2_bpf=900;  %设置解调器前端带通滤波器的通带长度
N_lpf=750;   %设置解调器的低通滤波器的截止频率
mt = Am*cos(2*pi*fm.*t); %调制信号
ct = cos(2*pi*fc.*t);    %载波
st = mt.*ct;             %已调信号
mt_spectrum = fft(mt,N)*ts; %调制信号傅立叶变换
st_spectrum = fft(st,N)*ts; %已调信号傅立叶变换
f = fs/2*linspace(0,1,N/2+1); %频率轴标注
```

【注解】

表 1　变量声明

变量名称	变量符号	变量值
调制信号频率	f_m	3kHz
载波频率	f_c	90kHz
采样频率	f_s	900kHz
数据序列长度	*L*	900
FFT 分析的点数	*N*	65536
调制信号	*mt*	/
载波	*ct*	/
已调信号	*st*	/
带通滤波器参数	*N1_bpf/N1_bpf*	通带 85kHz~95kHz
低通滤波器参数	*N_lpf*	截止频率 10kHz

Main 函数的第一部分首先进行了仿真参数的初始化，仿真时长为 1ms 的一段信号，共采集 900 个数据点。采样频率满足奈奎斯特定理。

2.2 DSB-SC 调制的时域和频域分析

```
%------------------------------------------------------------------------
%作图
%------------------------------------------------------------------------
--
figure(1);
subplot(2,1,1);
plot(t,mt,'-b');        %作图：调制信号
hold on
plot(t,st,'g');         %作图：已调信号
hold on
legend('调制信号','已调信号');
grid
```

```
xlabel('\itt/s','FontName','Times New Roman','FontSize',12);
ylabel('\itm(t)/s(t)','FontName','Times New Roman','FontSize',12);
title('调制信号与已调信号的波形','FontName','Times New
Roman','FontSize',12);

subplot(2,1,2);
plot(f,2*abs(mt_spectrum(1:N/2+1)),'-b')
hold on
plot(f,2*abs(st_spectrum(1:N/2+1)),'g')
hold on
legend('调制信号','已调信号');
grid
xlabel('\itf/\rmHz','FontName','Times New Roman','FontSize',12);
ylabel('\itM(jw/S(jw)','FontName','Times New Roman','FontSize',12);
title('调制信号与已调信号的频谱','FontName','Times New
Roman','FontSize',12);
```

【注解】

该段程序绘出了 DSB-SC 调制前后的时域及频域图形，如下图 3 所示

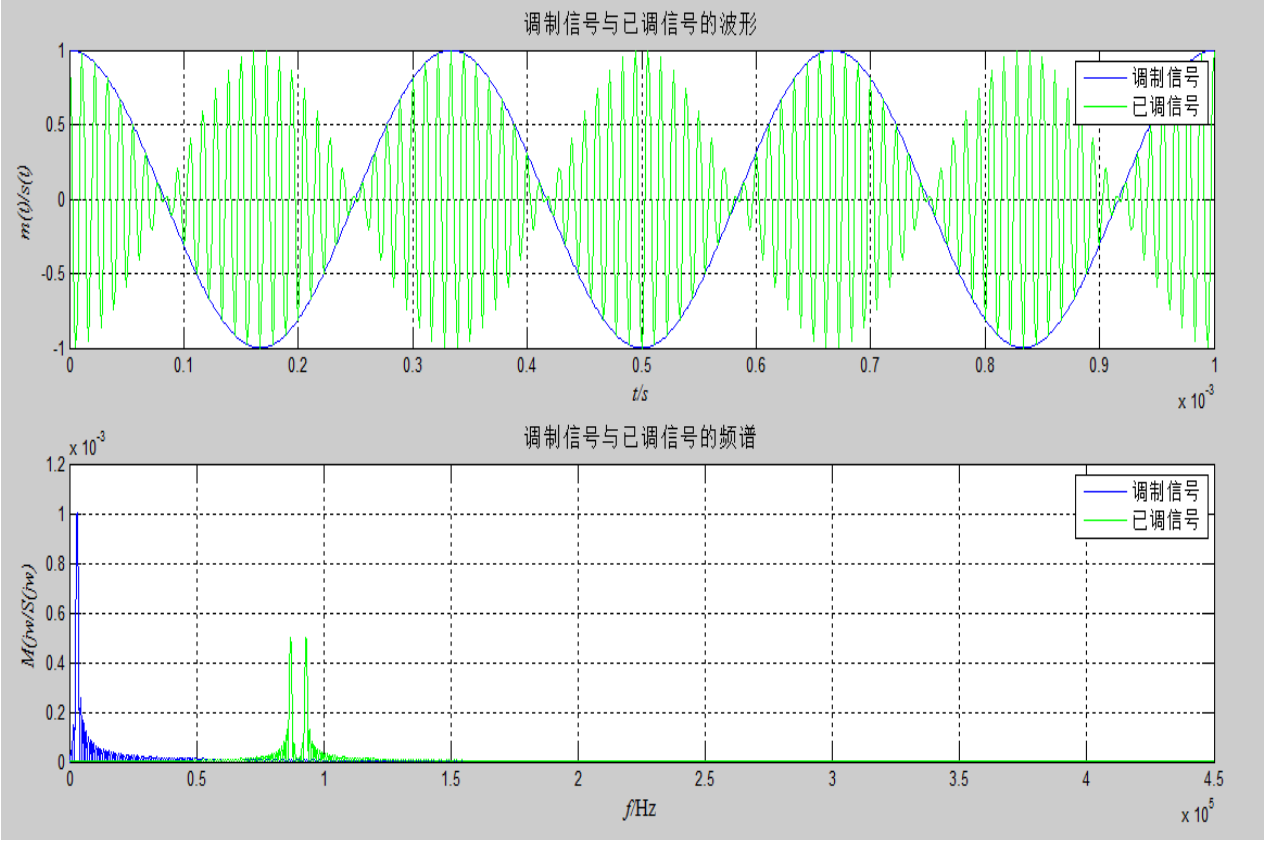

图 3　调制前后的时域及频谱图示

可以看出，调制后的时域波形包络反映了被调信号的信息，频谱也由原来的基带搬移到了关于载波频率对称的双边带上。

2.3 调制信号经过信道

```
%------------------------------------------------------------------------
%经过信道 叠加噪声
%------------------------------------------------------------------------
SNR1 = -5;   %信道信噪比-5dB经过带通滤波器后对应输入信噪比约为10dB
SNR2 = -15;  %信道信噪比-15dB经过带通滤波器后对应输入信噪比约为0dB
st_channel1 = add_noise(st,SNR1); %信道加噪声
st_channel2 = add_noise(st,SNR2); %信道加噪声
```

【注解】

已调信号经过信道后，叠加了白噪声。由于信道信噪比不直接等于输入信噪比（带通滤波器的作用），经过分析，信道信噪比分别为-5dB 和-15dB 时对应输入信噪比为 10dB 和 0dB。这里使用了 add_noise 函数加噪，函数的内容已在子文件 add_noise.m 中给出，添加于附录（下面的 bandpass_filter 和 lowpass_filter 亦同）。

2.4 DSB-SC 调制信号的解调

```
%------------------------------------------------------------------------
%解调端
%------------------------------------------------------------------------
st_channel1_input = bandpass_filter(N1_bpf,N2_bpf,N,L,st_channel1);
%理想带通滤波器的频域参数
bandpass_filter(N1_bpf,N2_bpf,N,L,st_channel2);
%理想带通滤波器的频域参数

%乘上载波进行相干解调
st_channel1_predemodulation = st_channel1_input.*ct;
st_channel2_predemodulation = st_channel2_input.*ct;

st_channel1_demodulation =
lowpass_filter(N_lpf,N,L,st_channel1_predemodulation); %理想低通滤波器参数
st_channel2_demodulation =
lowpass_filter(N_lpf,N,L,st_channel2_predemodulation); %理想低通滤波器参数

%------------------------------------------------------------------------
%作图
%------------------------------------------------------------------------
figure(2);
subplot(2,3,1);
plot(t,st,'-b');
hold on
grid
title('原已调信号\its(t)','FontName','Times New Roman','FontSize',12);
xlabel('\itt/s','FontName','Times New Roman','FontSize',12);
```

```
subplot(2,3,2);
plot(t,st_channel1_input,'r');
hold on
grid
title('信噪比10dB解调输入','FontName','Times New Roman','FontSize',12);
xlabel('\itt/s','FontName','Times New Roman','FontSize',12);

subplot(2,3,3);
plot(t,st_channel2_input,'k');
hold on
grid
title('信噪比0dB解调输入','FontName','Times New Roman','FontSize',12);
xlabel('\itt/s','FontName','Times New Roman','FontSize',12);

subplot(2,3,4);
plot(t,mt,'-b');
hold on
grid
title('原调制信号','FontName','Times New Roman','FontSize',12);
xlabel('\itt/s','FontName','Times New Roman','FontSize',12);

subplot(2,3,5);
plot(t,st_channel1_demodulation,'r');
hold on
grid
title('信噪比10dB解调输出','FontName','Times New Roman','FontSize',12);
xlabel('\itt/s','FontName','Times New Roman','FontSize',12);

subplot(2,3,6);
plot(t,st_channel2_demodulation,'k');
hold on
grid
title('信噪比0dB解调输出','FontName','Times New Roman','FontSize',12);
xlabel('\itt/s','FontName','Times New Roman','FontSize',12);
```

【注解】

调制信号在 10dB 和 0dB 的输入信噪比条件下，得到的解调输入及输出波形如下图 4 所示

可以看出， 输入信噪比 0dB 和 10dB 的解调波形相比无噪声时的情况都出现了不同程度的失真，其中前者更为明显。

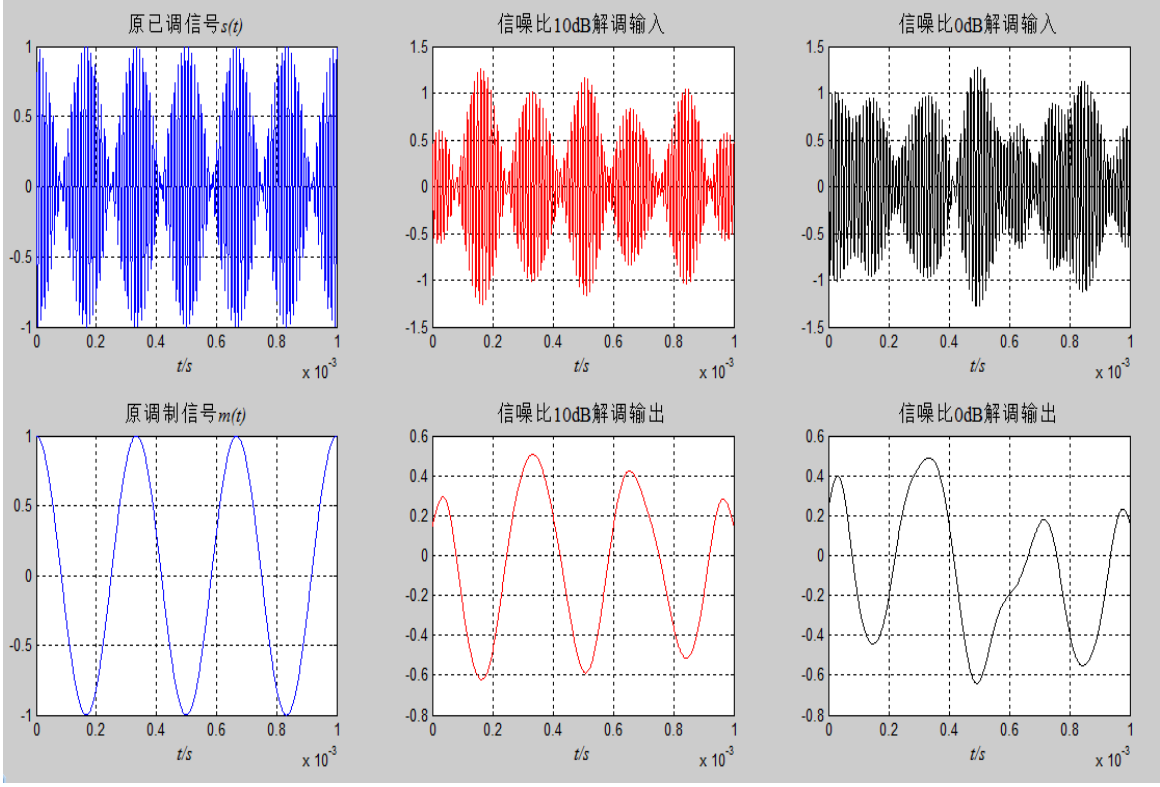

图 4　输入信噪比 0dB 和 10dB 的解调波形

2.5 输入信噪比与输出信噪比的关系

```
figure(3);
len=100; %信道信噪比变化范围
step=1;   %输入信噪比每次增加的步长
input = 0; %记录输入信噪比计算平均值
output = 0;% 记录输出信噪比计算平均值
SNR_input = zeros(1,len);  %建立保存输入信噪比数据的数组
SNR_output = zeros(1,len); %建立保存输出信噪比数据的数组

for SNR= step:step:len
st_channel = add_noise(st,SNR); %信道加噪
st_channel_input = bandpass_filter(N1_bpf,N2_bpf,N,L,st_channel);
%加噪信号通过解调器前端的带通滤波器
st_input = bandpass_filter(N1_bpf,N2_bpf,N,L,st);
%未加噪信号通过解调器前端的带通滤波器
%计算输入信噪比
SNR_input(SNR) = SNR_cal(st_input,st_channel_input);
%加噪信号的解调
st_channel_predemodulation = st_channel_input.*ct;
st_channel_demodulation =
lowpass_filter(N_lpf,N,L,st_channel_predemodulation);
%未加噪信号的解调
st_predemodulation = st_input.*ct;
```

```
st_demodulation = lowpass_filter(N_lpf,N,L,st_predemodulation);
%计算输出信噪比
SNR_output(SNR) = SNR_cal(st_demodulation,st_channel_demodulation);
%描点作图
plot(SNR_input(SNR),SNR_output(SNR),'*b');
hold on
end
%求平均的制度增益
gain = mean(SNR_output)-mean(SNR_input);
disp(['计算平均制度增益为：',num2str(gain),'dB']);
%disp(gain);
%½øÐÐÏßÐÔÄâºÏ
b = polyfit(SNR_input,SNR_output,1);
y = polyval(b,SNR_input);
plot(SNR_input,y,'k');
hold on
grid
xlabel('输入信噪比(dB)','FontName','Times New Roman','FontSize',12);
ylabel('输出信噪比(dB)','FontName','Times New Roman','FontSize',12);
title('输入信噪比与输出信噪比的关系曲线');
```

【注解】

令信道信噪比从 1dB 到 100dB 进行扫描，步长 1dB，经过带通滤波器后计算出输入信噪比，再由相干解调后得到输出信噪比。描点得到输出信噪比和输出信噪比关系的散点图，再进行线性回归分析。

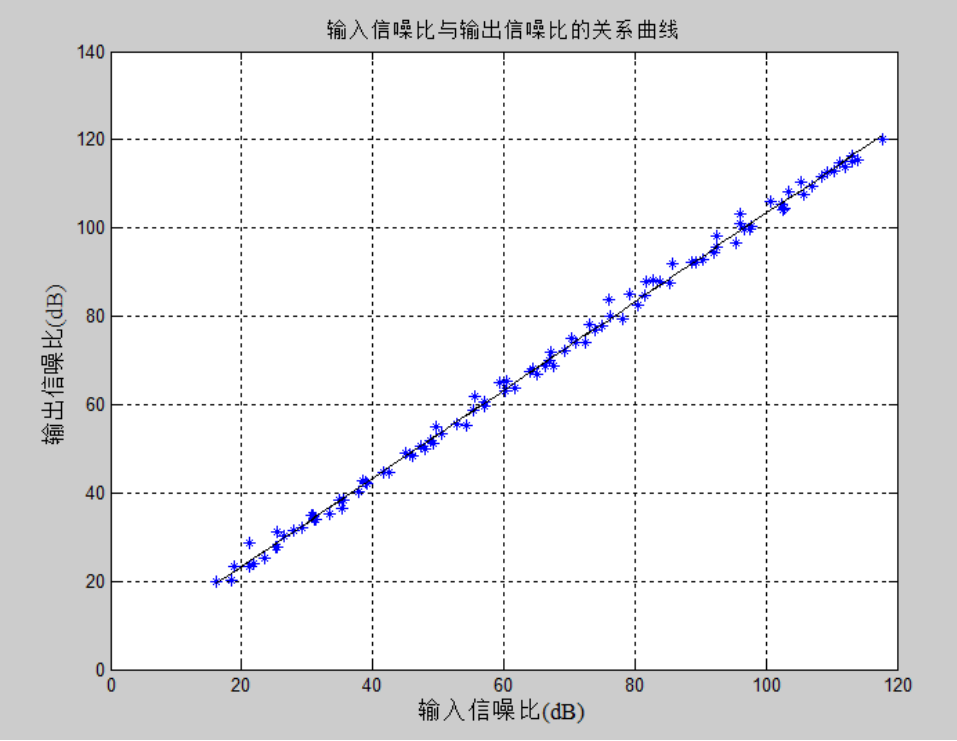

图 4　输入信噪比与输出信噪比的关系图示

可以看出，每组对应的输入信噪比和输出信噪比都在回归直线上下小区间内波动，由程序计算出 100 组数据的平均信噪比增益为 3.0123dB，基本与理论的制度增益 3dB 吻合。

（三）结论与收获

经过对于这一系列关于 DSB-SC 调制解调的 MATLAB 仿真，我对它的原理和过程有了更深入的了解和认识，仿真的结果与理论上的结论基本一致，这个验证的过程既是对已有结果的重现，也同时开启了对更多问题的思考。虽然在课上学习 DSB-SC 调制解调的时候以为自己已经听得非常明白了，但是到自己动手仿真这整个系统的时候才会注意到很多细节问题，这些问题可能不单单涉及到通信原理的内容，还可能与数字信号处理等等的学科相关。经过仿真才会慢慢注意到它们之间内在的一致联系。

附录

Main 函数中用到的几个函数：

bandpass_filter.m

```
function [ output ] =
bandpass_filter( N1_bandpass_filter,N2_bandpass_filter,N_FFT,data_lengt
h,input)
%在频域上设计理想带通滤波器，把信号频带滤出来
input_spectrum = fft(input,N_FFT);
bandpass_filter1 = zeros(1,N1_bandpass_filter);
bandpass_filter2 = ones(1,N2_bandpass_filter);
bandpass_filter3 = zeros(1,N_FFT/2 - N1_bandpass_filter -
N2_bandpass_filter);
bandpass_filter0 =
[bandpass_filter1,bandpass_filter2,bandpass_filter3,bandpass_filter3,ba
ndpass_filter2,bandpass_filter1];
%接收信号经过带通滤波器后的频谱
output_spectrum = input_spectrum.*bandpass_filter0;
%接收信号经过带通滤波器后的时域采样
output_spectrum_ifft = ifft(output_spectrum,N_FFT);
output = real(output_spectrum_ifft(1:round(data_length)));
end
```

lowpass_filter.m

```
function [ output ] =
lowpass_filter(N_lowpass_filter,N_FFT,data_length,input)
```

```
%在频域上设计理想低通滤波器
lowpass_filter1 = ones(1,N_lowpass_filter);
lowpass_filter2 = zeros(1,N_FFT/2 - N_lowpass_filter);
lowpass_filter0 =
[lowpass_filter1,lowpass_filter2,lowpass_filter2,lowpass_filter1];
%输入信号的频谱
input_spectrum = fft(input,N_FFT);
%输入信号经过低通滤波器后的频谱
output_spectrum = input_spectrum.*lowpass_filter0;
%输入信号经过低通滤波器后的时域采样output_spectrum_ifft =
real(ifft(output_spectrum,N_FFT));
output = real(output_spectrum_ifft(1:round(data_length)));
end
```

SNR_cal.m

```
function [ snr ] = SNR_cal( signal,signal_channel )
Ps = mean(sum((signal).^2));  %信号功率
Pn = mean(sum((signal-signal_channel).^2)); %噪声功率
snr = 10*log10(Ps/Pn);
end
```

add_noise.m

```
function [ Y,noise ] = add_noise( X,SNR )
noise = randn(size(X));
noise = noise - mean(noise);
signal_power = 1/length(X)*sum(X.^2);
noise_variance = signal_power/(10^(SNR/10));
noise = sqrt(noise_variance)/std(noise)*noise;
Y = X + noise;
end
```