

实验 1

一、实验目的

更改 Mf 程序中 sigma 和 a 取值，观察结果的变化。

二、实验原理

生成脉冲发射后的回波信号，并将回波与噪声相加得到接收信号，利用 sum 函数计算接收信号与脉冲信号在不同时间的相关性，可认为当 sum 函数的结果最大时，对应时间的相关性最大，利用最大值的索引估计脉冲回波的延时。

三、实验代码

```
clc;
clear;
randn('state',1); %固定生成的随机信号
N=500; %时间轴长度
M=20; %脉冲宽度
N0=248; %实际目标延时
a=input('a='); %信号幅度
sigma=input('sigma='); %噪声幅度
echo_signal=zeros(N,1);
transmit_signal=a*ones(M,1); %脉冲信号
echo_signal(N0:(N0+M))=a; %生成回波信号
receive_signal=echo_signal+sigma.*randn(N,1); %将回波与噪声相加得到接收信号
correl=zeros(N,1);
for k=1:(N-M)
    correl(k)=sum(transmit_signal.*receive_signal(k:(k+M-1)));
end %计算接收信号与脉冲信号的相关性，存放在 correl 中
[cmax,ctime]=max(correl); %取出 correl 中最大值的索引
delay=ctime % ctime 是估计的延时
subplot(3,1,1);
plot(echo_signal);
title(['a=' num2str(a), ',sigma=' num2str(sigma), '的回波信号']);
axis([0 N 0 3]);
subplot(3,1,2);
plot(receive_signal);
title(['a=' num2str(a), ',sigma=' num2str(sigma), '的接收信号']);
subplot(3,1,3);
plot(correl);
text(delay,correl(delay),['(',num2str(delay),',',num2str(correl(delay)),')'])
); %在滤波输出图中标注出最大值及其时间
axis([0 N min(correl) max(correl)*1.1]);
```

```
title('相关滤波器的输出');
```

四、结果分析

当 $a=2$, $\sigma=1$ 时, 结果如图 1-1。

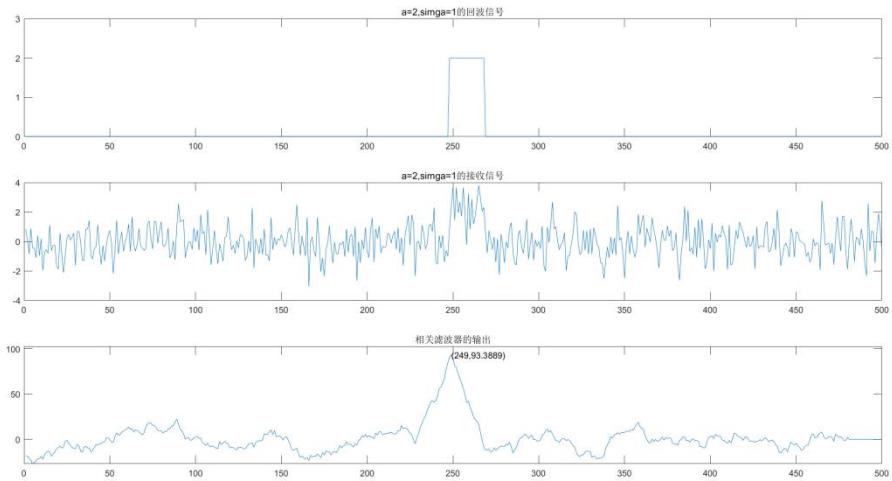


图 1-1

当 $a=2$, $\sigma=5$ 时, 结果如图 1-2。

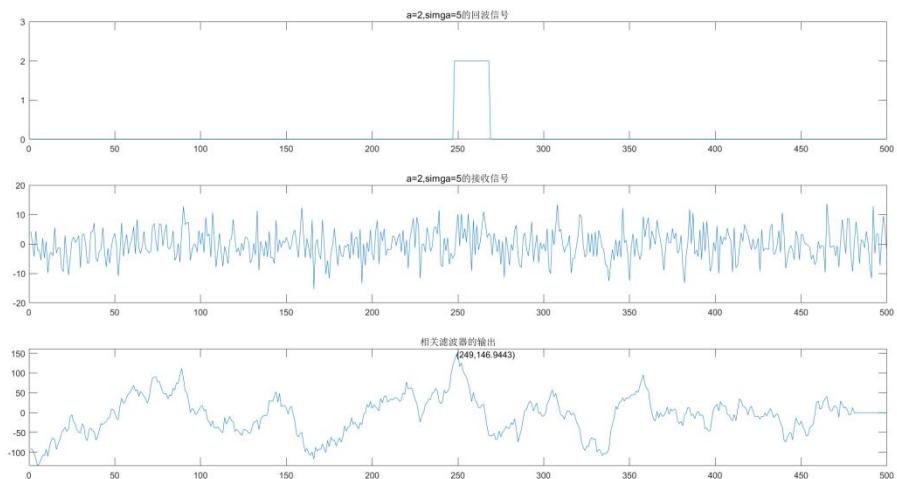


图 1-2

当 $a=2$, $\sigma=10$ 时, 结果如图 1-3。

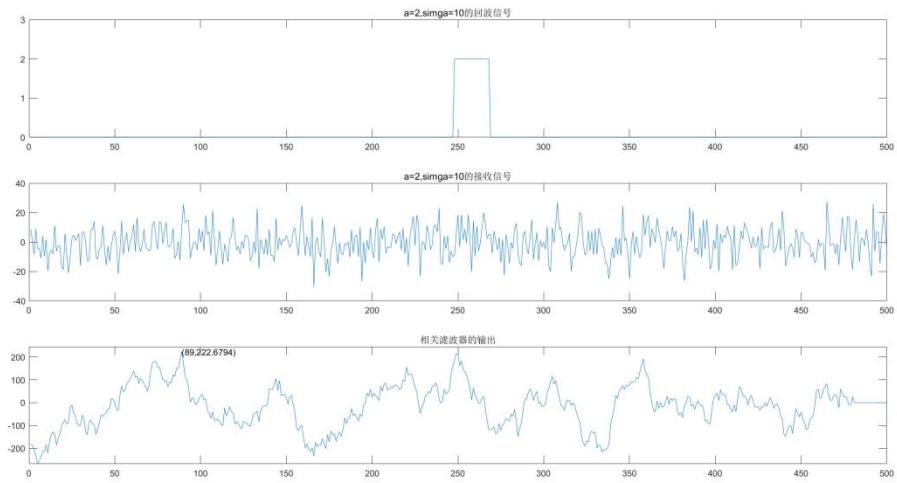


图 1-3

当 $a=10$, $\sigma=5$ 时, 结果如图 1-4。

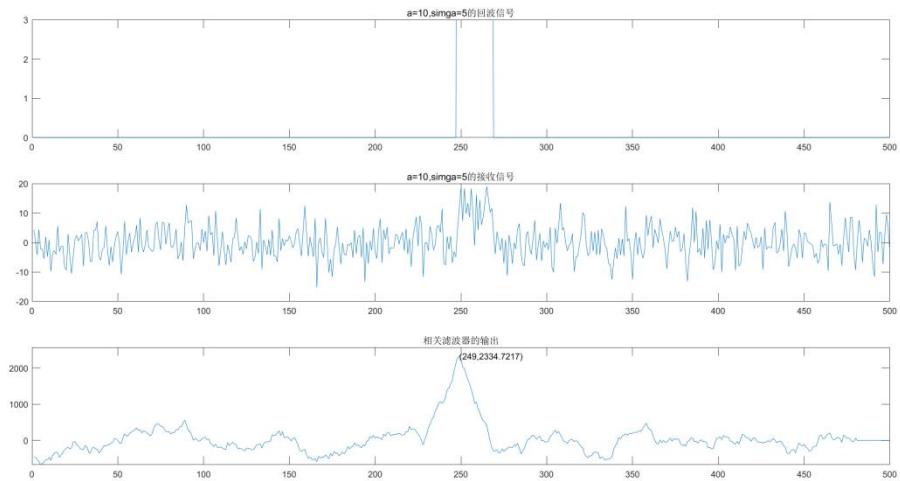


图 1-4

由图 1-1~图 1-3 可看出, 当 σ 增大时, 由于噪声幅度增大, 相关滤波器更容易在实际延时以外的时间点输出最大值。

而将图 1-4 分别与图 1-1、图 1-2 对比可知, 相关滤波器的输出波形几乎只与 a 和 σ 的比值相关, 信号与噪声幅度等比例变化并不会影响相关滤波器的输出波形。

实验 2

一、实验目的

更改 Mf.error 程序中 sigma 取值，观察均值和方差的变化。

二、实验原理

在实验 1 的基础上，进行多次仿真，每次仿真估计的结果都会与实际延时相减作为误差存入 d_e 矩阵中，利用矩阵的数据绘制图像，观察每次仿真的误差，并用 hist 函数统计各误差值出现的次数。

三、实验代码

```
clc;
clear;
randn('state',1);
N=500;
M=20;
MN=5000;    %蒙特卡洛仿真的次数
N0=248;
a=1.5;
sigma=input('sigma=');
echo_signal=zeros(N,1);
transmit_signal=a*ones(M,1);
echo_signal(N0:(N0+M-1))=a;
for j=1:MN
    receive_signal=echo_signal+sigma.*randn(N,1);
    correl=zeros(N,1);
    for k=1:(N-M)
        correl(k)=sum(transmit_signal.*receive_signal(k:(k+M-1)));
    end
    [cmax,ctime]=max(correl);
    delayestimate_error(j)=ctime-N0;    %将估计延时与实际延时的误差存入 d_e 矩阵中
end
subplot(231);
plot(echo_signal);
title(['a=' num2str(a),',sigma=' num2str(sigma), '的回波信号(最后一次仿真)']);
axis([0 N 0 2*a]);
subplot(234);
plot(receive_signal);
title(['a=' num2str(a),',sigma=' num2str(sigma), '的接收信号(最后一次仿真)']);
subplot(232);
plot(correl);
```

```

axis([0 N -100 150]);
title('相关滤波器的输出');
subplot(235);
plot(delayestimate_error);
title('延时估计的误差');
subplot(133);
histogram(delayestimate_error); %用 hist 函数统计各误差值出现的次数
title(['simga=' num2str(sigma), '时各误差值出现的次数']);

```

四、结果分析

由实验 1 结论可知，相关滤波器的输出波形由 a 与 σ 的比值决定，故此处选定 $a=1.5$ ，更改 σ 的取值，观察现象。

当 $\sigma=0.8$ 时，仿真结果如图 2-1。

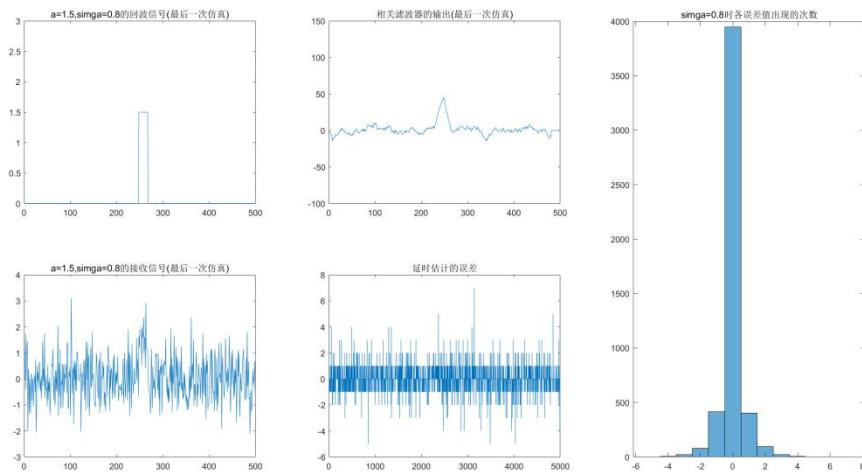


图 2-1

当 $\sigma=1$ 时，仿真结果如图 2-2。

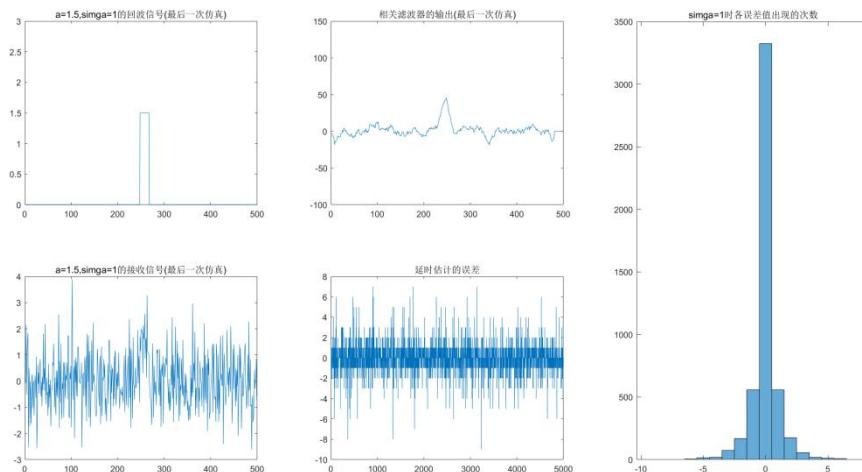


图 2-2

当 $\sigma = 1.2$ 时，仿真结果如图 2-3。

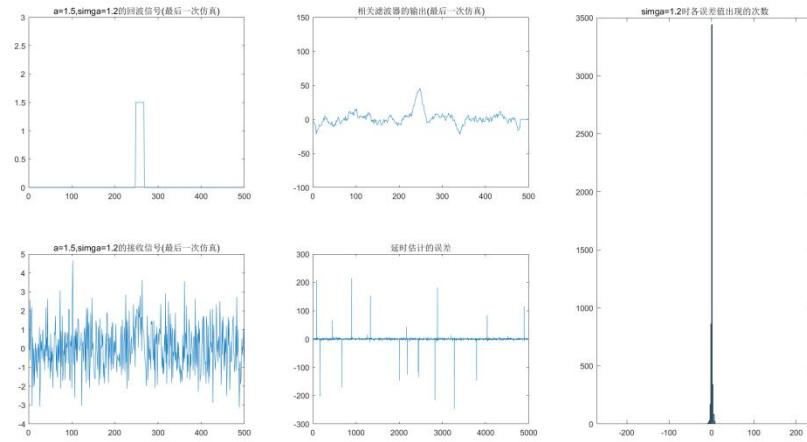


图 2-3

当 $\sigma = 1.5$ 时，仿真结果如图 2-4。

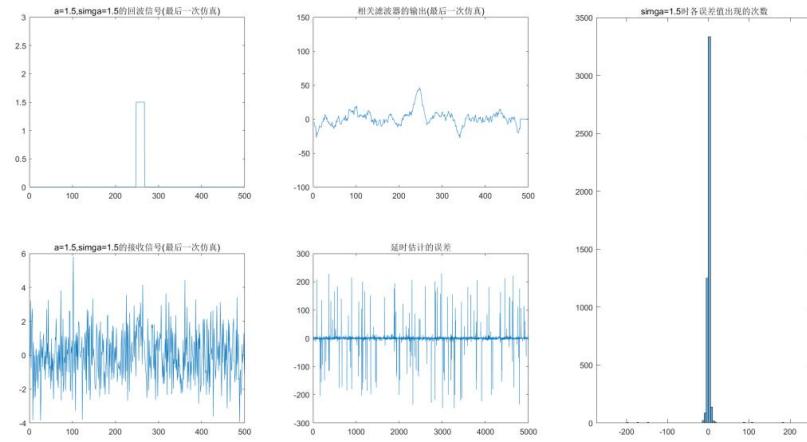


图 2-4

为了更直观体现均值和方差与 σ 的关系，绘制曲线如图 2-5。

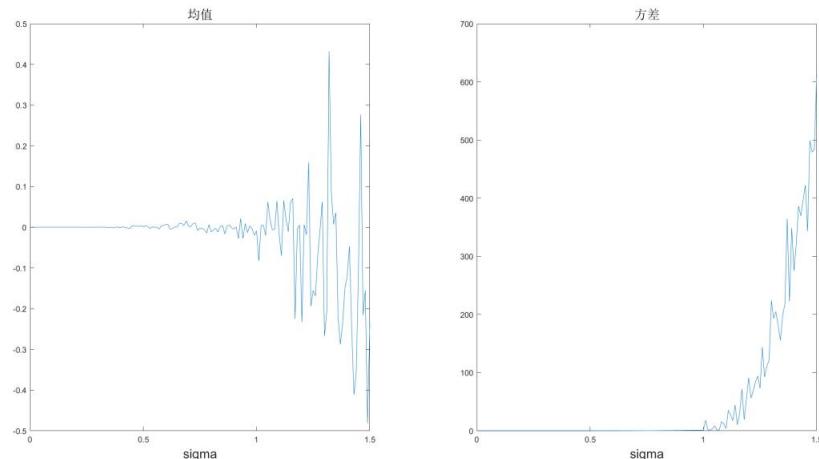


图 2-5

由图 2-5 可看出, 当噪声幅度越来越大, 均值的绝对值和方差也呈增长趋势, 但均值一直在 0 上下浮动, 且浮动范围并不大, 而方差在 σ 大于 1 时呈现飞快的增长趋势。

结合图 2-3、2-4 可推测, 这种现象是因为, 当噪声幅值达到一定高度时, 接收信号已经不稳定, 滤波输出的结果可能在其他时间出现最大值, 而此时刻与实际延时相差巨大, 导致某几次仿真出现巨大误差。由图 2-3、2-4 的滤波输出可看到, 最后一次仿真的估计值与实际值相差了将近 150 个时间单位。由条形图的横轴也可看出, $\sigma=0.8$ 和 1 时, 基本只出现过 10 以内的误差, 而 $\sigma=1.2$ 和 1.5 时, 坐标轴已经扩展到了几百, 说明此时已经开始出现巨大误差。由方差计算的定义, 方差值为误差的平方级, 所以当 σ 大于 1 时, 方差随着 σ 的增大而急剧增长。

当继续增大 σ , 得到方差变化趋势如图 2-6。

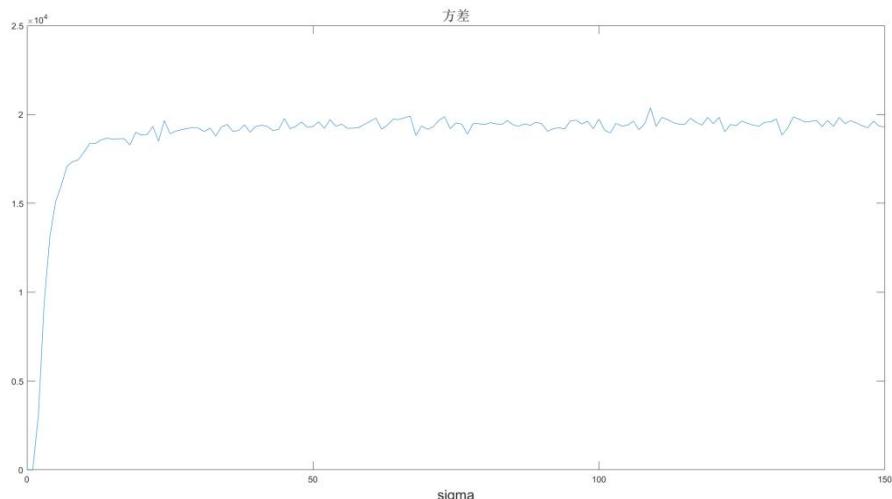


图 2-6

由图可知, σ 大于 10 之后, 方差已经趋于平稳。结合前面的推测可知, 这是因为仿真的时间轴长度只有 500, 脉冲延时为 248, 当 σ 大于 10, 即使继续增大, 每次仿真的误差也只会落在 -247~252 之间。误差有上限, 方差自然也会有上限, 故呈现出图 2-6 所示趋势。

实验 3

一、实验目的

分析 BP_pulse 程序，理解脉冲压缩的作用及原理。

二、实验原理

实验生成的长度为 N 的 BK 码和 m 序列满足以下条件

1. 序列中的元素只有 1 或 -1。

2. 序列中的元素共轭反转后的序列与原信号进行卷积，若两者波形不对齐，每个时间点的乘积结果都是 -1 或 1，卷积结果为 ±1 或 0；而两者波形对齐时，每个时间点的结果都是 $-1 \times (-1)$ 或 1×1 ，卷积结果为 N。于是卷积得到的序列中，只有两序列对齐时的索引处卷积结果为 N，其余索引处的卷积结果均为 0 或 1。

将满足条件的 BK 码或 m 序列与噪声信号相加得到回波信号，将回波信号反转的 bp_MF 序列与回波序列作卷积。在这个过程中，噪声成分被叠加或抵消，增益并不明显；而 BK 码或 m 序列成分在目标时延处的卷积结果得到了明显增益，时域带宽也非常窄，从而实现脉冲压缩。

三、实验代码

```
clc;
clear;
BP_Type = input('BP_Type=');
BP_Length = 511;    % 63/127/255/511/1023/2047
noieseAmp = input('noieseAmp=');  % 噪声幅度
bp_pc_result = BP_Pulse_Comp(BP_Type, BP_Length, noieseAmp);
function bp_pc_result = BP_Pulse_Comp(BP_Type, BP_Length, noieseAmp)
if BP_Type == 1
    bp_echo = [1,1,1,1,1,-1,-1,1,1,-1,1,-1,1];      % 生成 BK 码
else
    bp_echo = produce_m(BP_Length);      % 生成 m 序列
end
sig_length = length(bp_echo);
noise1 = noieseAmp * randn(1, sig_length);      % 生成噪声信号
bp_echo_noise = bp_echo + noise1;      % 回波信号
subplot(221);
plot(bp_echo);
if BP_Type == 1
    title('BK 码回波信号')
else
```

```

    title('m 序列回波信号')
end
axis([1 length(bp_echo) -2 2]);
subplot(222);
plot(bp_echo_noise);
%chirp_MF: chirp 脉压的匹配滤波器，信号的(共轭)反转
bp_MF = bp_echo(length(bp_echo) : -1 : 1);      %将回波信号反转
title('加噪声的回波信号');
axis([1 length(bp_echo_noise) min(bp_echo_noise)-1 max(bp_echo_noise)+1]);
%二相码脉冲压缩
bp_pc_result = conv(bp_echo_noise, bp_MF);
[cm,ct]=max(bp_pc_result);
subplot(223);
plot(bp_pc_result);
text(ct,cm,['(',num2str(ct),',',',',num2str(cm),')']);
title('二相码脉压的结果');
xlim([1,length(bp_pc_result)]);
bp_pc_result = abs(bp_pc_result) + 0.1;      %加 0.1 防止对 0 求对数
subplot(224);
plot(20*log10(bp_pc_result/max(bp_pc_result)));
title('二相码脉压的结果(dB 表示)');
ylabel('dB');
xlim([1,length(bp_pc_result)]);
end
%产生 m 序列的函数
function ms = produce_m(ms_Length)
if ms_Length == 63
    x=[0,0,1,0,1,0];    %basic code, 可以设置为任何不为全 0 的 0-1 序列
    y=[5,6];    %feedback link
elseif ms_Length == 127
    x=[0,0,1,0,1,0,0];
    y=[6,7];
elseif ms_Length == 255
    x=[0,0,1,0,1,0,0,0];
    y=[4,5,6,8];
elseif ms_Length == 511
    x=[0,0,1,0,1,0,0,0,1];
    y=[5,9];
elseif ms_Length == 1023
    x=[0,0,1,0,1,0,0,0,1,0];
    y=[7,10];
elseif ms_Length == 2047
    x=[0,0,1,0,1,0,0,0,1,0,1];
    y=[9,11];

```

```

end
k=length(x);
ms=x;
if ms_Length == 255
    for i=k+1:2^k-1
        tmp1 = xor( ms(i-y(1)), ms(i-y(2)) );
        tmp2 = xor( ms(i-y(3)), tmp1 );
        tmp3 = xor( ms(i-y(4)), tmp2 );
        ms(i) = tmp3;
    end
else
    for i=k+1:2^k-1
        ms(i)=xor(ms(i-y(1)),ms(i-y(2)));
    end
end
ms=ms*-2+1;      %将非 0 即 1 序列转化为非-1 即 1 序列
end

```

四、结果分析

由代码可知, produce_m 函数中利用一系列的“异或”运算生成满足脉冲压缩条件的 m 序列, 而 bp_pc_result 函数中的 BK 码也满足这一条件。

先生成一段长 13 的 BK 码, 将噪声幅度设为 1, 得到图 3-1。

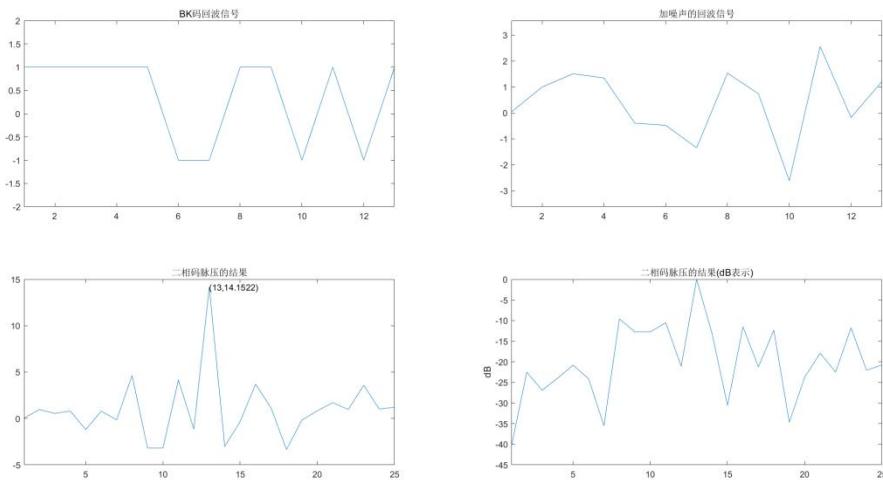


图 3-1

接着将噪声幅度提高至 5, 得到图 3-2。

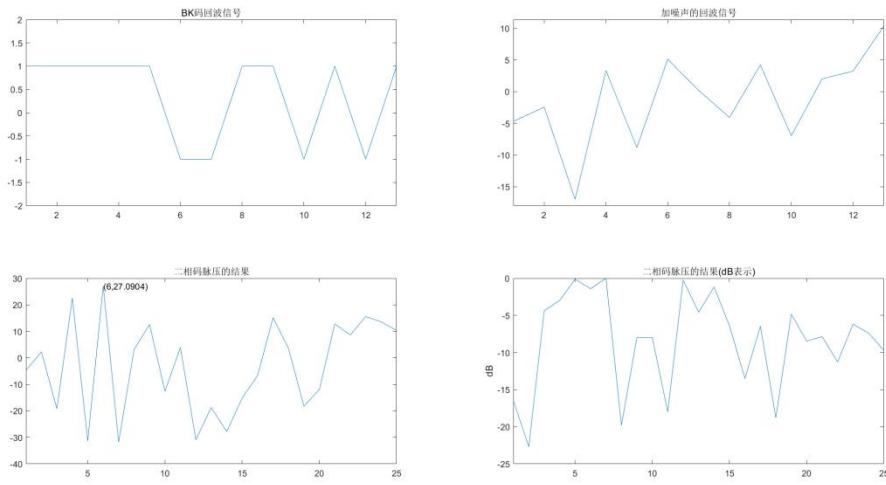


图 3-2

保持噪声幅度为 5, 将 BP_Type 设为 2, 即生成长 255 的 m 序列, 得到图 3-3。

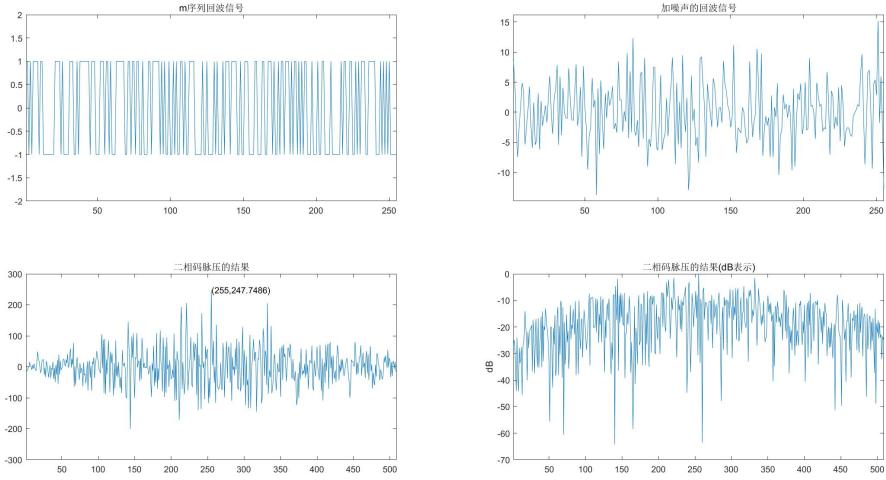


图 3-3

再将噪声幅度提高到 10, 得到图 3-4。

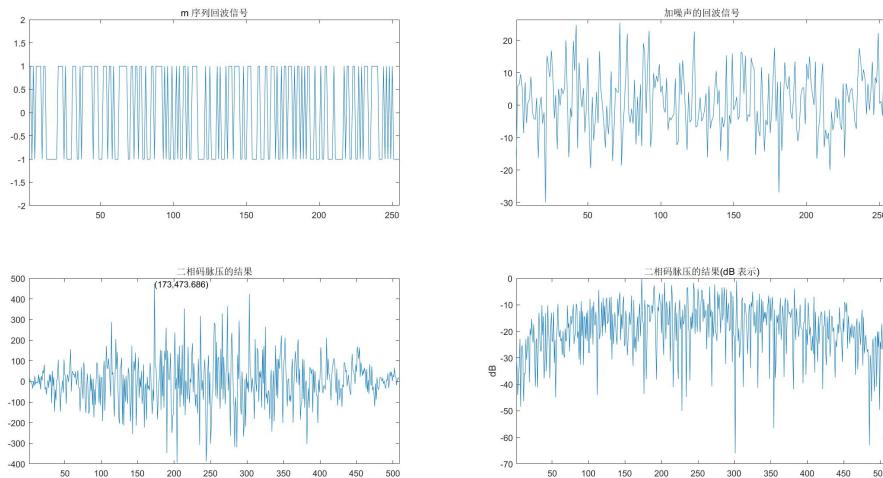


图 3-4

此时将 BP_length 改成 511，得到图 3-5。

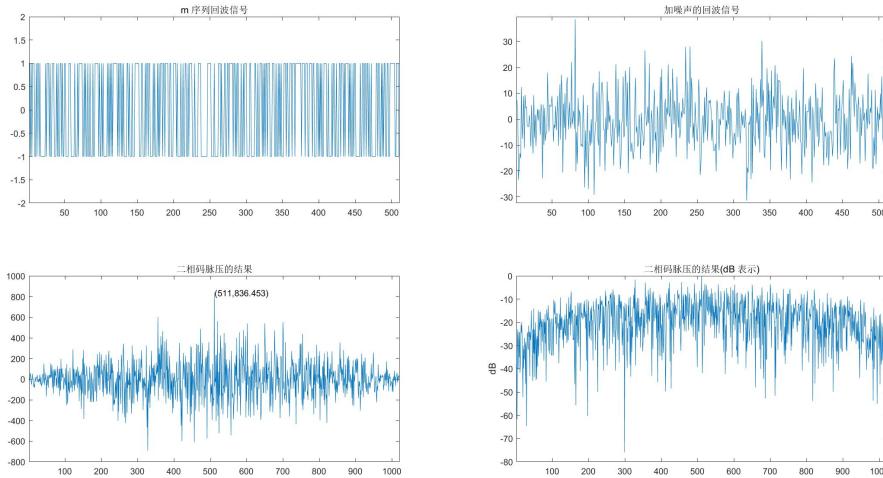


图 3-5

由 3-1~3-2 和 3-3~3-4 两组图对比可看出，信号长度固定，当噪声幅度达到一定值，卷积的结果就可能出现偏差，这是因为噪声的基础幅度远大于信号时，即使卷积的增益不如信号，增益后的幅值也可能超过脉冲压缩的结果。

而通过 3-2~3-3 和 3-4~3-5 两组图及后续多次测试，可以看出，信号的发射时间越长，其发射功率越大，信噪比越高。

将实验 3 与实验 2 的一系列结果对比可知，如果使用实验 2 的简单方波信号进行卷积，即使噪声非常微弱，卷积后的结果也是在实际时延的位置形成一个三角波，导致时域带宽向两侧严重扩散。当噪声幅度增大，卷积结果的最大值就很

容易出现在附近的其他时间点，导致时延估计不准确。此外，如果有两个邻近的大小目标，卷积后的大三角波可能会将小三角波覆盖，导致目标测量出现偏差。

而实验 3 中利用 BK 码和 m 序列进行脉冲压缩，大大削减了卷积结果的时域带宽，使信号非常集中，提高了信噪比，更能避免使用方波带来的一系列问题。