天津大学

无线计算器实验报告

学　　院　电气自动化与信息工程

专　　业　　通信工程

年　　级　　2018 级

姓　　名　　苏一牧、孟昶旭

学　　号 3018234436、3018234431

2020 年 12 月 18 日

1. 实验要求

（1）在‘发送电脑’上输入一个整截四则运算式（+、-、*、/），运算式中的数字为 0~10000，输入完成后，这个算式通过耳麦被传到‘接受电脑’上，‘接受电脑’对算式进行计算，计算的结果同样通过耳麦返回到‘发送电脑’上，‘发送电脑’对此接收后显示结果。

（2）在算式输入完毕后 1 秒钟内要将结果显示出来。如果在传输过程中出现错误，要能自动重传，重传 N 次，最长等待时间 N 秒。（由于误码率很低，没有做重传系统）

（3）两个耳麦的距离为 10cm（实际实验情况为 2cm 左右）

（4）传输过程中有一个人为添加的噪声，这个噪声的强度很大，但是频带在一定的范围内。（这里选取了 4KHz 和 2KHz，因此可以抗人为语音干扰）

2. 实验环境

（1）两台标准个人电脑；

（2）两个低质量耳麦，分别连接到一台电脑上；

（3）操作系统：Windows10；

（4）应用软件：Matlab;

（5）硬件设置：控制面板>声音>播放>扬声器>高级---------关闭音频增强，设置 DVD 音质，16 位，48000HZ；

控制面板>声音>录制>麦克风>高级>----------关闭音频增强，DVD 音质，16 位，48000HZ；

控制面板>声音>录制>麦克风>级别>----------麦克风加强 30db，麦克风 100；

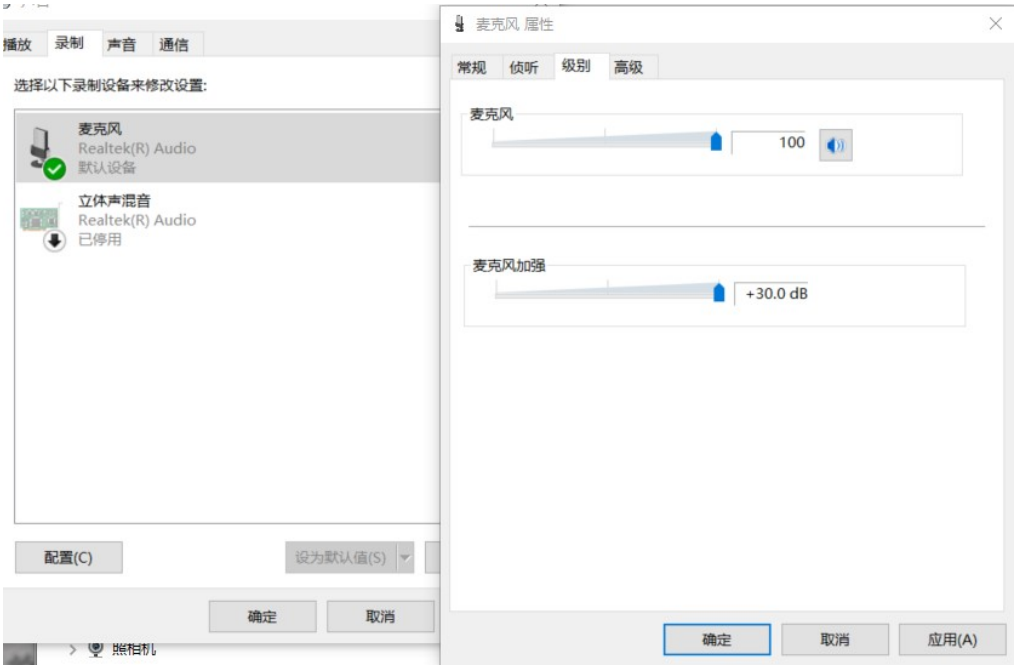

图 1 声卡设置

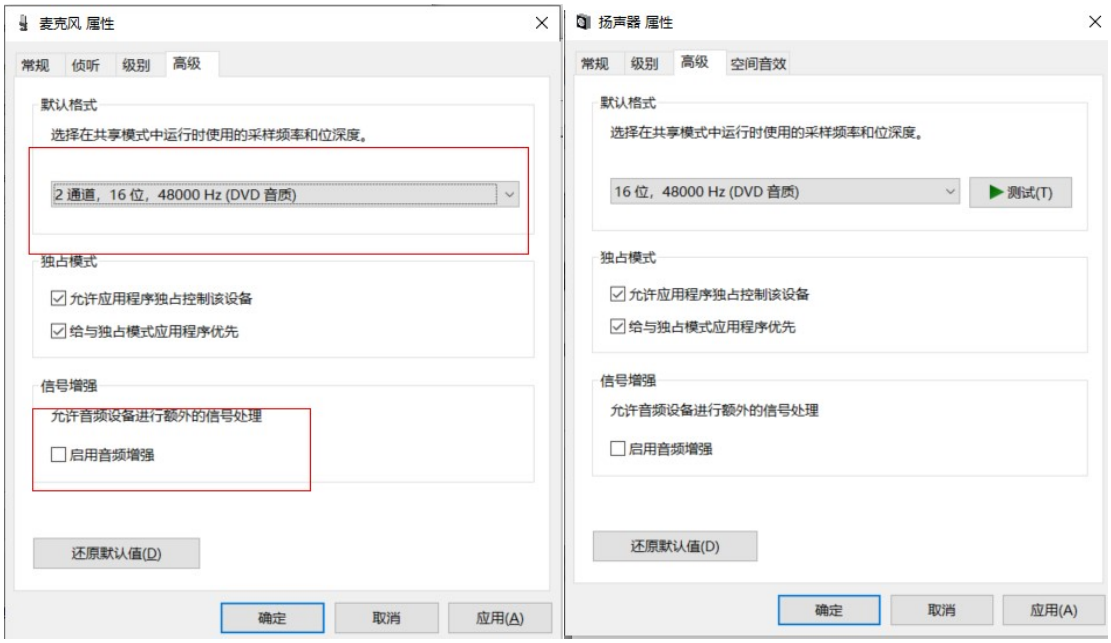

图 2 采样率设置

备注：此步是在进行两台电脑音卡的同步，一定要关闭音频增强，音频增强会自动滤除一些高频的音频部分，即对输入或者输出的声音信号进行额外的处理，这种黑盒会给声音通信系统的设计造成很大的麻烦。

3. 设计过程

系统简介：本系统利用了 2FSK 的调制方式，有以下特点：

（1）增加了巴克码帧同步以及抽样判决前的积分过程，使得整体的传输速率大大提高，传信率可以达到 200b/s（误码率极小），理论上可以达到 400b/s。

（2）应用了（7,4）汉明码，可以实现一位自动纠错。

（3）采用了归一化设计，所有过程都封装成了函数，方便调用。参数可以随意调节，因此很容易就可以实现频分复用。

（4）为了调制后两载波主瓣不重叠，载波频率相距越远越好，考虑到相位连续，载波频率最好是在成比例的偶数倍，普通计算机在设计时就考虑到人声频率，能量最大值在 0~4000Hz 之间（因此建议选择 3000Hz~6000Hz），而在 1000Hz 时能量虽然最大，但人声和物体在这个频段内干扰也极其严重，所以我们最终选择 2000Hz，4000Hz 作为最后的载波频率。

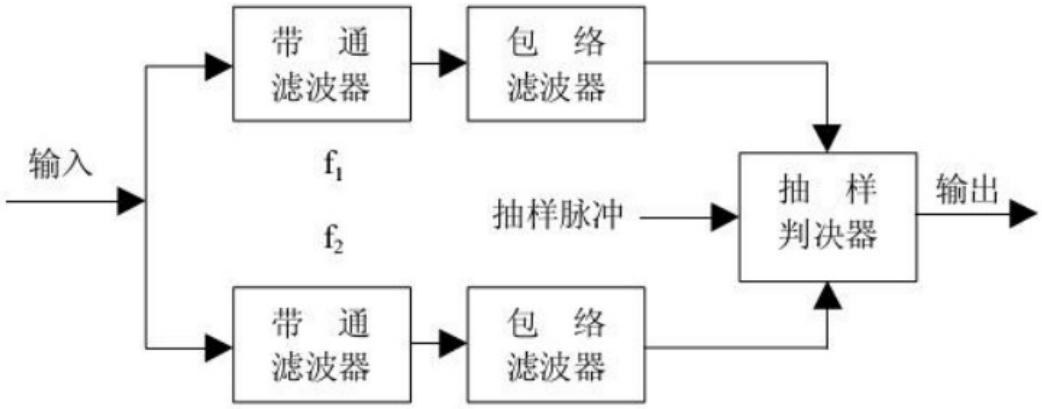

图 3 2FSK 解调框图

系统总体框图如下：

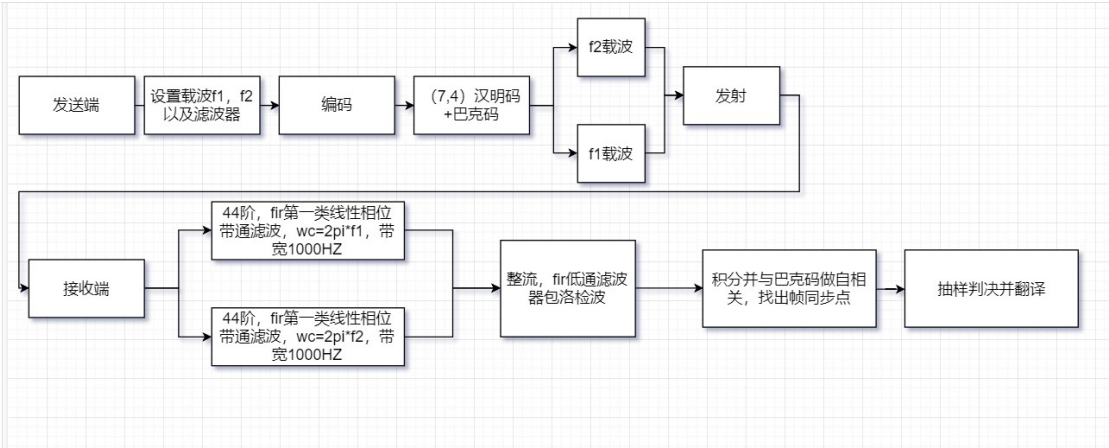

图 4 系统总体框图

相关技术参数：f1=200HZ，f2=4000HZ，低通 fp=160hz。

编码后将原码拓宽了 160 倍，fs=32000；

采用 13 位巴克码 BAKA=[1 1 1 1 1 0 0 1 1 0 1 0 1]。

应用了该技术参数，最后实现了 在 1min15s 内，发送并回传 100 次算式，成功率 100%。

4. 代码详解

所有代码可以浓缩成以下几个函数：

```
%--------------------------------------------------------------------写好的函数
function [pyr1, f3, f2, f1]=jianbo(bp2000, bp1000, lpf, aaaa, pyr)...
function[b3]=baka(b, N)...
function [str]=fanyi(b3)...
function [F] = yima(LCD_str)...
function [bittosend_final]= tuokuan(bbbb, N)...
function [modulatedData]= zaibo(f1, f2, fs, c)...
```

4.1 编码/译码（yima）

4.1.1 信源编码

因为无线计算器涉及到的符号最少有 16 个，为最大化传输速率，在信源编码上我们采用了 4bit 来表示所需的符号

```
case '+'
    bittosend_temp0=[bittosend_temp0 '0000'];
case '-'
    bittosend_temp0=[bittosend_temp0 '0001'];
case '*'
    bittosend_temp0=[bittosend_temp0 '0010'];
case '/'
    bittosend_temp0=[bittosend_temp0 '0011'];
case '0'
    bittosend_temp0=[bittosend_temp0 '0100'];
case '1'
    bittosend_temp0=[bittosend_temp0 '0101'];
case '2'
    bittosend_temp0=[bittosend_temp0 '0110'];
case '3'
    bittosend_temp0=[bittosend_temp0 '0111'];
case '4'
    bittosend_temp0=[bittosend_temp0 '1000'];
case '5'
    bittosend_temp0=[bittosend_temp0 '1001'];
case '6'
    bittosend_temp0=[bittosend_temp0 '1010'];
case '7'
    bittosend_temp0=[bittosend_temp0 '1011'];
case '8'
    bittosend_temp0=[bittosend_temp0 '1100'];
case '9'
    bittosend_temp0=[bittosend_temp0 '1101'];
case '('
    bittosend_temp0=[bittosend_temp0 '1110'];
case ')'
    bittosend_temp0=[bittosend_temp0 '1111'];
otherwise
    disp('输入数据有误');
```

4.1.2 信道编码

我们采用了（7,4）汉明码进行了信道编码，虽然会增加 3 位监督位，影响码率，但其带来了纠错能力的提升。

```
bittosend_temp2=str2num(bittosend_temp0');%将字符串转换成数字型
b2=encode(bittosend_temp2,7,4,'hamming/binary');
%增加前面的保护码，同步吗巴克码
```

4.1.3 帧同步与保护

为了明确码元的抽样时刻和有效信息的提取，我们在信道编码后又增加了自相关函数具有尖锐单峰特性的巴克码作为帧同步码，因其伪同步的可能性很小，故在后面进行时分复用时仍有极好的鲁棒性。同时加入保护码是因为在不同性能的计算机上能量识别的速度不同，保护码的长度可以很好的调整这一参数。

```
F=[ones(200,1);[1 1 1 1 1 0 0 1 1 0 1 0 1]';b2;zeros(11,1);[1 1 1 1 1 0 0 1 1 0 1 0 1]';];
```

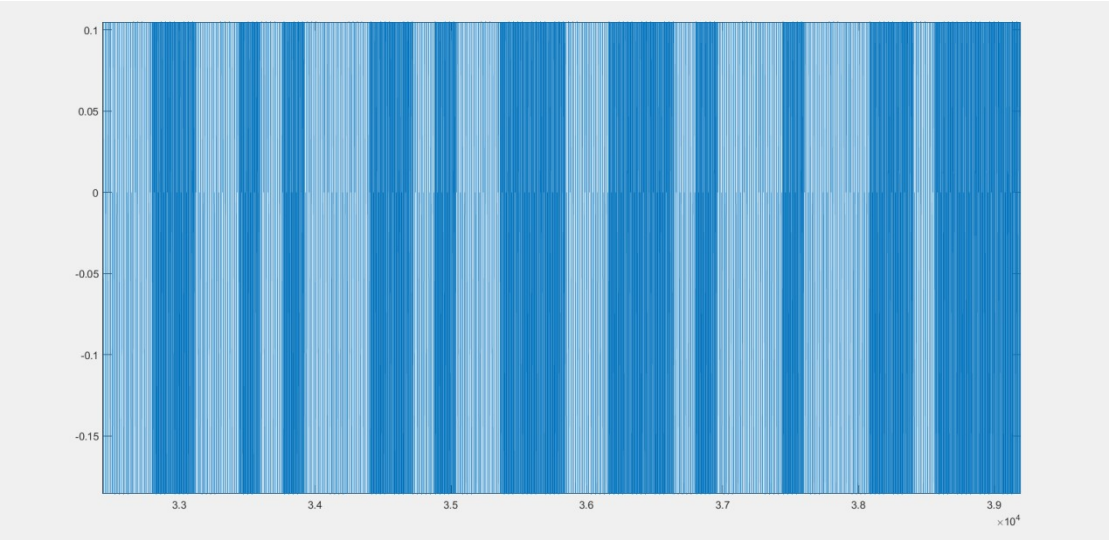

图 5 编码后产生的最终效果

4.2 巴克码（baka）

在抽样判决后，我们将码元去掉前后的巴克码，再进行汉明解码，最终将结果再翻译成原有信息进行计算，因为我们采用了时分复用，算式之间用‘/’隔开，因此在接收时识别到‘/’后就将算式分开计算。

```
function[b3]=baka(b,N)
 bark=[1 1 1 1 1 0 0 1 1 0 1 0 1];
 for index=1:length(b)-12
        res_xor=xor(b(index:index+12),bark)
        if(sum(res_xor))<N
            break;
        end
 end
for i=index+13:length(b)-12
        res_xor=xor(b(i:i+12),bark)
        if(sum(res_xor))<N
            break;
        end
end
disp(i);
disp(index);
b3=b(1,index+13:i-11);
end
```

4.3 拓宽（tuokuan）

一般来说，因受计算机性能限制，展宽越宽，准确度越高，速度越小，我们最终经过尝试，将码元展宽 160 倍，码元传输速率 R_B=32000/160=200bps，将函数封装后可以更方便更改参数。

```
function [bittosend_final]= tuokuan(bbbb,N)
        bittosend_final=[];
        for index=1:length(bbbb)
            if bbbb(index)
                bittosend_final=[bittosend_final ones(1,N)];
            else
                bittosend_final=[bittosend_final zeros(1,N)];
            end
        end
end
```

4.4 载波（zaibo）

因 MATLAB 计算矩阵的便捷快速，利用矩阵乘法将调制后码元与载波结合，将载波按其对应频率间隔取样，载波 1 重复 10 次，载波 2 重复 10*f1/f2 次，将信息调制完成。

```
function [modulatedData]= zaibo(f1,f2,fs,c)
t2=1/(fs/f2)/f2:1/(fs/f2)/f2:1/f2;
t1=1/(fs/f1)/f1:1/(fs/f1)/f1:1/f1;
carrier1Data_temp=sin(2*pi*f2*t2);
carrier1Data=repmat(carrier1Data_temp,1,10)

carrier2Data_temp=sin(2*pi*f1*t1);
carrier2Data=repmat(carrier2Data_temp,1,10*f1/f2);%
modulatedData=[];
for index=1:100:length(c)
    if c(index)==1
        modulatedData=[modulatedData carrier1Data];
    else
        modulatedData=[modulatedData carrier2Data];
    end
end
end
```

4.5 判断接收开始

接收端先进行一步能量检测，能量检测时同样进行滤波处理，因环境因素影响需根据图像设置判决门限，每隔 0.5s 取一段信息进行比较，当检测到能量突发时进行声音录制。

```
while(flag==0)
    voicing = audiorecorder(fs,16,1);
    recordblocking(voicing, 0.5);
    voice = getaudiodata(voicing);
    voice=filter(bp1000,1,voice);
    plot(voice);
    disp('正在检测，开始检测中.....')
    if max(voice) > 0.2
        flag=1;
        disp('已接通，开始接收信息…')
    end
end
```

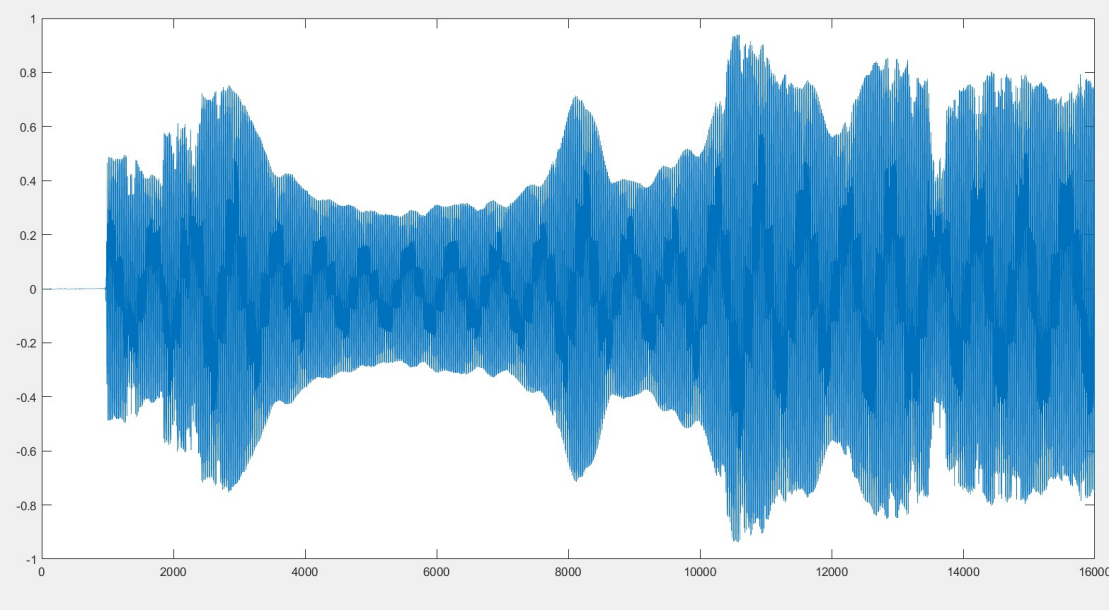

图 6 能量检测识别成功效果图

4.6 滤波并求包络（jianbo）

首先根据要求设计带通滤波器，经过尝试，我们最终选择了 44 阶 fir1 带通滤波器，中心频率选择 2000Hz，4000Hz 归一化频率，通带带宽选择 1000Hz。包络检波器采用 fir1 低通滤波，通带带宽 160Hz。

```
%------------------构建滤波器
bp2000=fir1(44,[f11/fss f12/fss]);
bp1000=fir1(44,[f21/fss f22/fss]);
lpf=fir1(44,160/fss);  %200HZ低通
```

下图为滤波器的实际效果（1,1）为载波 1 滤波后结果，（1,2）为载波 2 滤波后结果，（2,1）为对载波 1 进行包络检波后结果，（2,2）为对载波 2 进行包络检波后结果，因为这已经是最大码率下的结果，人眼识别不明显，但已起到了实际效果，（3,1）为相减后结果，判决门限为 0，最终实现了滤波器部分的设计与实现。

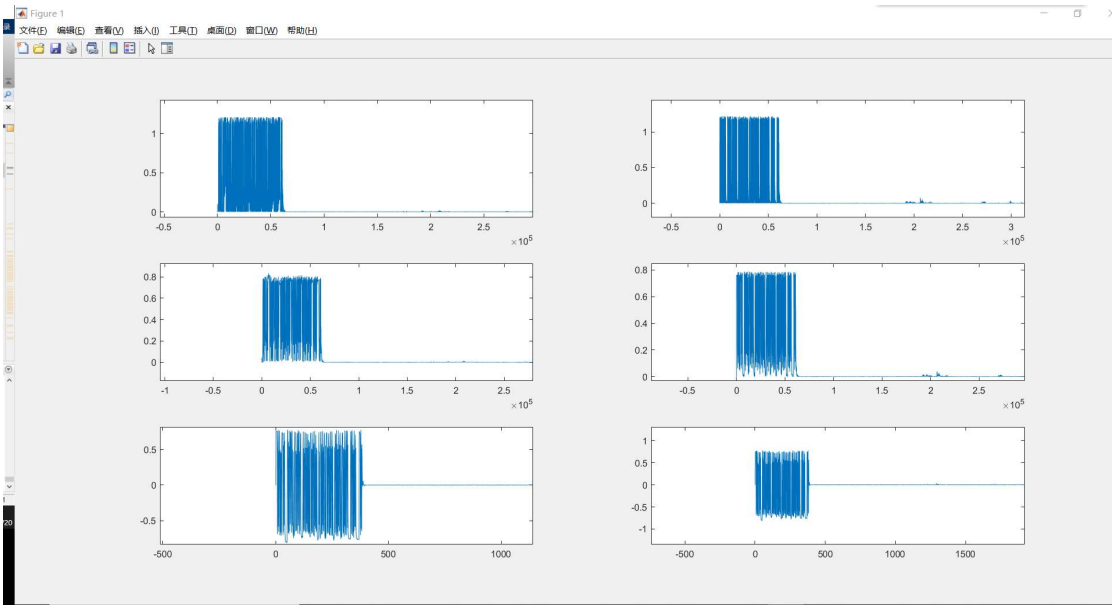

图 7 载波 1,2 经过滤波器和包络检波后的效果图

4.7 积分，抽样判决和自相关函数的求取

这一点是我们要着重介绍的部分，也是我们作改进的最大的一部分。其中的核心代码在以下：

```
pyr1=p(1:20000,1);%取出一段,认为这一段包含巴克码,找到巴克码大概在哪;
bark=[1 1 1 1 1 0 0 1 1 0 1 0 1];
[pyr3,~,~,~]=jianbo(bp2000,bp1000,lpf,DOWN,pyr1);
pyr3(pyr3>0)=1;
pyr3(pyr3<=0)=-1;
a1=pyr3;
bark1=[ones(1,5*DOWN) -ones(1,2*DOWN) ones(1,2*DOWN)  -ones(1,DOWN)  ones(1,DOWN) -ones(1,DOWN) ones(1,DOWN)];
a=0;
c=[];
for b=1:length(a1)-13*DOWN
    c(b)=bark1*a1(b:b+13*DOWN-1,1);
end
[~,u]=max(flip(c));
[~,v]=max(c);
u=length(c)+1-u;
z=floor((u+v)*0.5);
%-------------------------------------------------------------------算出帧同步点在哪
```

```
[pyr3,~,~,~]=jianbo(bp2000,bp1000,lpf,DOWN,pyr1);
```

此步完成了对于原始函数的滤波以及包洛检波

```
pyr3(pyr3>0)=1;
pyr3(pyr3<=0)=-1;
```

而我应用了以上两句话进行抽样判决，代替了 if 语句，使得计算速度有了一个质的提升。当对巴克码进行拓宽和转置之后，最精彩的部分就开始了：

```
for b=1:length(a1)-13*DOWN
    c(b)=bark1*a1(b:b+13*DOWN-1,1);
end
```

在这里，我们给 0 赋加权-1，给 1 赋加权 1，完成了矩阵乘法后就可以找到因变量 c 最大的那个点，而这个点便是我们想要的帧同步点。可以总结为下面三张图片：

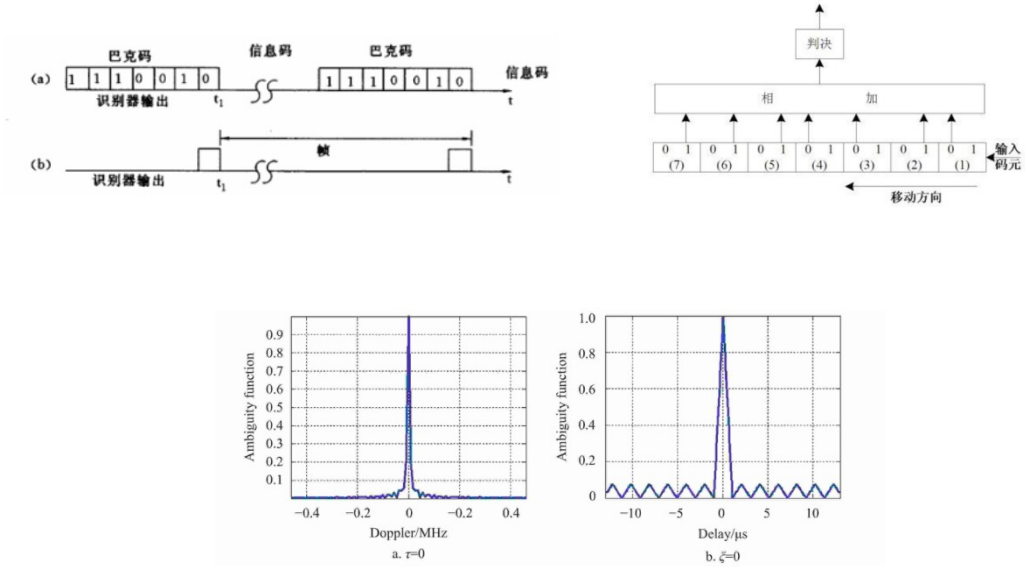

图 8 同步点查找示意图

如图所示，我们认为这个极大值点就是帧同步点：

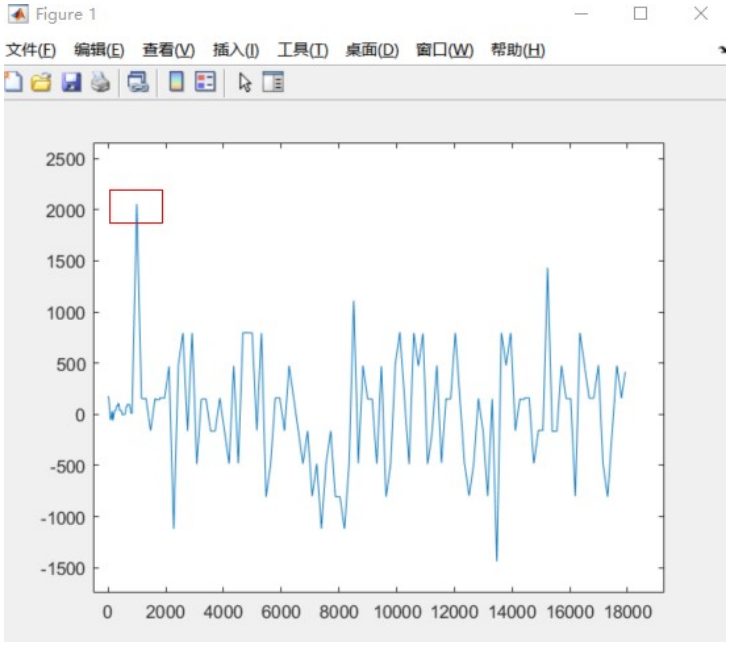

图 9 帧同步点查找结果

```
pyr666=p(z-40-DOWN:end,1)';
[a2,~,~,~]=jianbo(bp2000,bp1000,lpf,DOWN,pyr666);
a2=a2(1,1:floor(length(a2)/DOWN)*DOWN);

a2(a2>0)=1;
a2(a2<=0)=-1;
a2=reshape(a2,DOWN,[]);
a3=sum(a2);

a3(a3>0)=1;
a3(a3<0)=0;%------进行积分然后直接判决，都利用了进行运算；
```

之后就是在进行积分，然后判决，判决完之后进行翻译即可

5. 实验感想

（1）首先感谢侯老师的指导，在老师的指导下，我们少走了很多弯路；其次感谢王亚飞学长的代码，给我们具体实现一些功能提供了思路。

（2）经过这次实验，我们对于数字滤波器的使用有了更加深刻的认识，为了达到所需的要求，不仅需要理论上对滤波器进行设计，更需要在实际测试过程中，逐步画出对应图像，检验滤波性能进行修改。

（3）受到王亚飞学长代码的影响，我们最初版本的抽样判决部分应用了大量的 if 语句，对于整体的计算十分不利。电脑判断 2000 次，就需要 3s 钟，这也给我们更新算法带来了限制，一开始我们想法是先对所有函数进行归一化，之后频分复用。但是因为电脑发声能量分布不均，频分复用并不能大幅度提高时间，只是缓兵之计。最后我们发现了矩阵算法，所有的 if 语句都改用了矩阵算法后，大大提高了计算速度，也让求自相关函数成为了可能。

（4）实验中碰到的最大的困难就是电脑的声卡是有限制的，这个问题我们研究了好几天，最后发现只要设置好声卡相关参数，关闭音频自动增强就可以实现较为理想的信号传输。

（5）开始时，我们忽略了耳麦的条件，选择了 2FSK 的调制方式，不应用耳麦，直接利用电脑的耳机进行互传，依旧可以成功，说明 2FSK 的调制方式比较适用于变参信道。