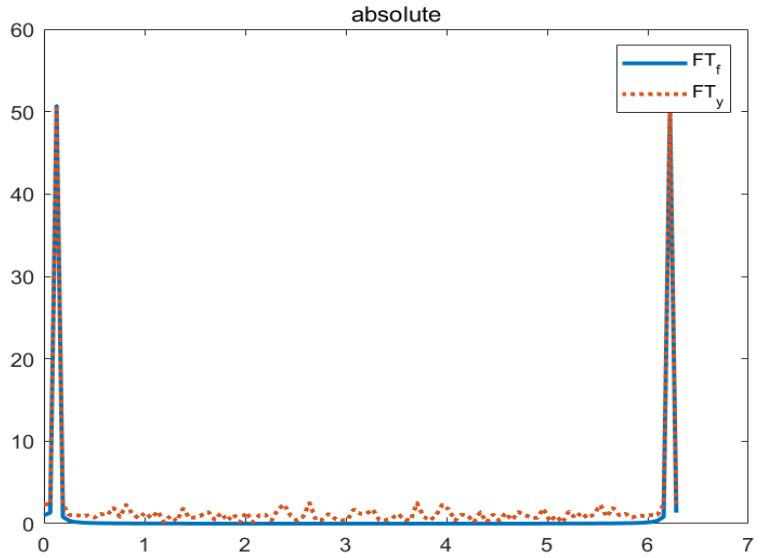

图 2: fft

结果分析：

可以看出来硬阈值法比均值滤波法更有效，这是因为原函数的傅里叶变换，两端模很大，而中间几乎为 0。

2 Q2

完成一半的风见鸡的合成，作品声音是某种合成的击弦音（本想合成钢琴音但放弃），代码及解释如下：

WindVine 分为两个声部，即 Rythem,Melody。下述代码为合成及写入命令。代码如下：

Listing 3: Code of WindVane

```
%% WindVane
%% Combine
fs=44100;

RythmPart=Rhythm();
MelodPart=Melody();
```

```
7
8
9  % RythmPart=0;
10 % MelodPart=0;
11
12
13 s=RythmPart+MelodPart;
14 s=s./max(s,2);
15
16 %% Experiment
17 sound(s,fs)
18
19 %% output
20 audiowrite('WindVane.mp4',s,fs)
```

Melody 和 Rythem 函数均为乐谱，其中通过传递乐器类型，音域给函数 Tone 获得不同音长的音高，下面代码以 Melody 为例：

Listing 4: Code of Melody

```
1  function MelodPart=Melody()
2  [F,h,fd,f,ed,e,s,¬,¬,fdblk,fblk,edblk,eblk,sblk]=Tone(@GuitarMe,3,7); ...
       % use guitar, tones are in [C3,C6)
3  F=@(n) F(n+7);
4  h=@(n) h(n+7);
5  fd=@(n) fd(n+7);
6  f=@(n) f(n+7);
7  ed=@(n) ed(n+7);
8  e=@(n) e(n+7);
9  s=@(n) s(n+7);
10 %% First Paragraph
11 p1=[e(7),s(5),e(7),e(7),e(8),e(5),f(8),sblk,...% sec
12      e(6),s(6),e(6),e(7),h(7),sblk,...% sec
13      e(7),s(5),e(7),e(7),e(8),e(5),f(8),sblk,...% sec
14      e(6),s(6),e(6),e(7),h(7),sblk,...% sec
15      ];
16 %% Paragraph 2
17 p2=[edblk,s(5),repmat(s(9),1,4),s(9),e(8),s(7),s(7),e(8),ed(7),...% sec
18      e(5),s(5),e(6),h(7),sblk,...% sec
19      edblk,s(5),repmat(s(9),1,4),s(9),e(8),s(7),s(7),e(8),ed(7),...% sec
20      e(5),s(5),e(6),h(5),sblk,...% sec
21      edblk,s(5),repmat(s(9),1,4),s(9),e(8),s(7),s(7),e(8),ed(7),...% sec
22      e(5),s(5),e(6),h(7),sblk,...% sec
23      edblk,s(5),repmat(s(9),1,4),s(9),e(8),s(7),s(7),e(8),ed(7),...% sec
24      e(5),s(5),e(6),h(5),sblk,...% sec
```

```
        ];
%% Paragraph 3
p3=[fblk,f(10),ed(10),ed(11),e(12),...%sec
        ed(11),ed(9),ed(9),sblk,e(9),e(8),e(7),...%sec
        fd(8),eblk,ed(8),ed(9),e(10),...%sec
        F(9),...%sec
        ];
%% Paragraph 4
p4=[...
    fblk,f(10),ed(10),ed(11),e(12),...%sec
    ed(11),s(14),sblk,s(13),sblk,fd(12),s(12),s(13),s(14),...%sec
    s(14),s(15),s(14),fd(12),sblk,fdblk...%sec
    f(11),f(12),f(13),e(12),e(13),...%sec
];
%% Paragraph Main1
p5=[...
        repmat(e(14),1,3),s(16),e(14),e(13),e(13),s(11),s(11),s(12),...%sec
        repmat(e(13),1,3),s(14),e(13),e(12),f(12),sblk,...%sec
        s(12),s(10),s(11),e(12),e(12),e(11),e(11),e(13),e(11),s(10),...%sec
        e(10),s(9),e(9),e(13),sblk,s(12),e(13),f(14),sblk,...%sec
];
%% Paragraph Main2
p6=[...
        repmat(e(14),1,3),s(16),e(14),e(13),e(13),s(11),s(11),s(12),...%sec
        repmat(e(13),1,3),s(14),e(13),e(12),f(12),sblk,...%sec
        s(12),s(10),s(11),e(12),e(12),h(11),sblk,...%sec
        f(9),f(10),h(11),...%sec
];
%% return
MelodPart=[
        1/4.*p1,...
        p2,...
        p3,...
        p4,...
        p5,...
        p6,...
];
end
```

Tone 函数根据给定乐器类型和音域生成音高数组，这一部分参考了嘉哥的 maryslamb

Listing 5: Code of Tone

```
function ...
```

```
[F,h,fd,f,ed,e,s,blkF,blkh,blkfd,blkf,blked,blke,blks]=Tone(Type,n,m)
%% Define Note
fs = 44100; % Standard sample rate
dt = 1/fs; % Standard sampling time interval
Beat=74; % 每分钟Beat拍
T16 = dt*(fs/Beat*15-1); %To determine the time length of a 1/16 ...
    note, suggest as an odd number
t16 = 0:dt:T16;
[¬,k] = size(t16);
t1 = linspace(0,16*T16,16*k);% An array with the same length as a full ...
    note
t2 = linspace(0,8*T16,8*k);
t4d = linspace(0,6*T16,6*k);%A special array represents a 1/4+1/8 note
t4 = linspace(0,4*T16,4*k);
t8d = linspace(0,3*T16,3*k);%A special array represents a 1/8+1/16 note
t8 = linspace(0,2*T16,2*k);
%% Define Frequency
f0 = 440/2^(5/12); % C3 tone
f0=f0*2^((n-3)*7/12); % Cn tone
ScaleTable = (2^(1/12).^(0:7*(m-n)-1))';%Other frequencies in [Cn,Cm)

        % full notes
[F,blkF]=Type(ScaleTable.*f0,t1);
F=@(n) F(n,:);

        % 1/2 notes
[h,blkh]=Type(ScaleTable.*f0,t2);
h=@(n) h(n,:);

% 1/4+1/8 notes
[fd,blkfd]=Type(ScaleTable.*f0,t4d);
fd=@(n) fd(n,:);

% 1/4 notes
[f,blkf]=Type(ScaleTable.*f0,t4);
f=@(n) f(n,:);

% 1/8+1/16 notes
[ed,blked]=Type(ScaleTable.*f0,t8d);
ed=@(n) ed(n,:);

% 1/8 notes
[e,blke]=Type(ScaleTable.*f0,t8);
e=@(n) e(n,:);

% 1/16 notes
```

```
45  [s,blks]=Type(ScaleTable.*f0,t16);
46  s=@(n) s(n,:);
47
48
49  end
```

乐器函数有两个 GuitarMe 和 GuitarRy，下面代码以 GuitarMe 为例。输入为音高和时长，输出即为声音。波形函数为多次尝试得到的多个三角函数叠加（这里其实还可以再调整以找到更好的波形），振幅（包络线）函数根据 $\frac{t^a}{e^{kt}}$ 调整参数 a 和 k 得到，由此得到某种合成的击弦音，如下：

Listing 6: Code of GuitarMe

```
1  function [soundGuit,blk] = GuitarMe(f,t)
2  % 该函数周期为2，改变频率*2f
3  harm=-1./4.*sin(3.*pi.*2.*f.*t)+1./2.*sin(2.*pi.*2.*f.*t)+...
4      1./4.*sin(pi.*2.*f.*t)+sqrt(3)./2.*cos(pi.*2.*f.*t);
5
6  a=0.5*1e-3;k=5;
7  A=t.^a./exp(k.*t);
8
9  soundGuit=A.*harm;
10
11 soundGuit=soundGuit./max(soundGuit,2);
12
13 blk = zeros(size(A));
14 end
```

输出结果得到 1 分 30 秒的合成击弦音的风见鸡

3 Q3

使用暴力（BF）算法匹配前 3 个音乐片段和 5 个乐曲。（不适用 KMP 算法，因重复部分极少）

BF 代码如下：

Listing 7: Code of BF

```
1  function id=BF(S,T)
2  ls=length(S);lt=length(T);
3  for i=1:ls-lt+1
```

```
    i_temp=i;
    flag=true;
    for j=1:lt
        if norm(S(i_temp,:)-T(j,:))>1e-4
            flag=false;
            break;
        end
        i_temp=i_temp+1;
    end
    if flag
        id=i;
        return;
    end
end
id=0;
end
```

接下来使用 BF 函数匹配 3 个片段，代码如下：

Listing 8: Code of Q3(1)

```
%% read
fs=44100;
f1=audioread('Music1.mp3');
f2=audioread('Music2.mp3');
f3=audioread('Music3.mp3');
f4=audioread('Music4.mp3');
f5=audioread('Music5.mp3');
fp1=audioread('musicpiece1.flac');
fp2=audioread('musicpiece2.flac');
fp3=audioread('musicpiece3.flac');
fp4=audioread('musicpiece4.flac');
fp5=audioread('musicpiece5.flac');
%% (1)
% piece 1
id11=BF(f1,fp1)
id12=BF(f2,fp1)
id13=BF(f3,fp1)
id14=BF(f4,fp1)
id15=BF(f5,fp1)

% piece 2
id21=BF(f1,fp2)
id22=BF(f2,fp2)
id23=BF(f3,fp2)
id24=BF(f4,fp2)
```

```
26 id25=BF(f5,fp2)
27
28 % piece 3
29 id31=BF(f1,fp3)
30 id32=BF(f2,fp3)
31 id33=BF(f3,fp3)
32 id34=BF(f4,fp3)
33 id35=BF(f5,fp3)
```

输出结果如下：

Listing 9: Outcomes of Q3

```
1  id11 =
2       0
3  id12 =
4       2337301
5  id13 =
6       0
7  id14 =
8       0
9  id15 =
10      0
11 id21 =
12      0
13 id22 =
14      0
15 id23 =
16      0
17 id24 =
18      0
19 id25 =
20      8070301
21 id31 =
22      0
23 id32 =
24      0
25 id33 =
26      0
27 id34 =
28      2734201
29 id35 =
30      0
```

结果分析：

片段 1 与第 2 个乐曲匹配，匹配采样点为 2337301，而采样率为 44100，即匹配点为 53 秒。

片段 2 与第 5 个乐曲匹配，匹配采样点为 8070301，而采样率为 44100，即匹配点为 183 秒。

片段 1 与第 2 个乐曲匹配，匹配采样点为 2734201，而采样率为 44100，即匹配点为 62 秒。

作业文件完。