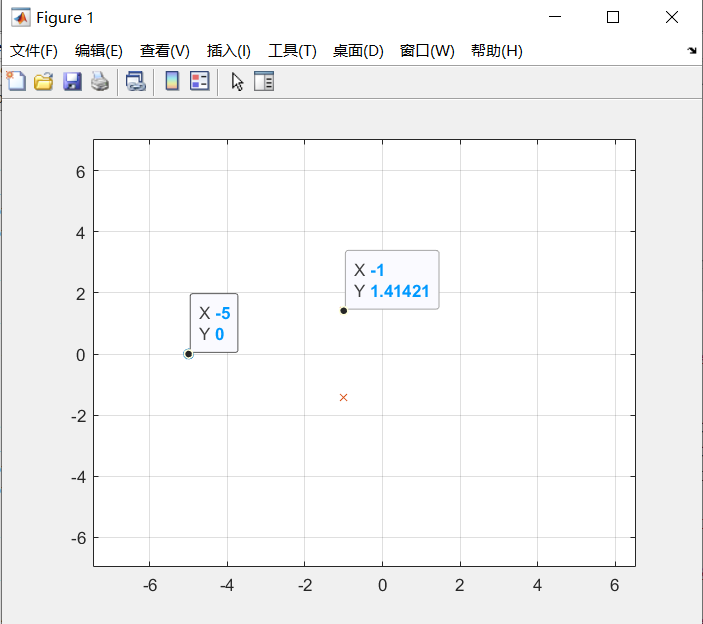
# project 1 说明

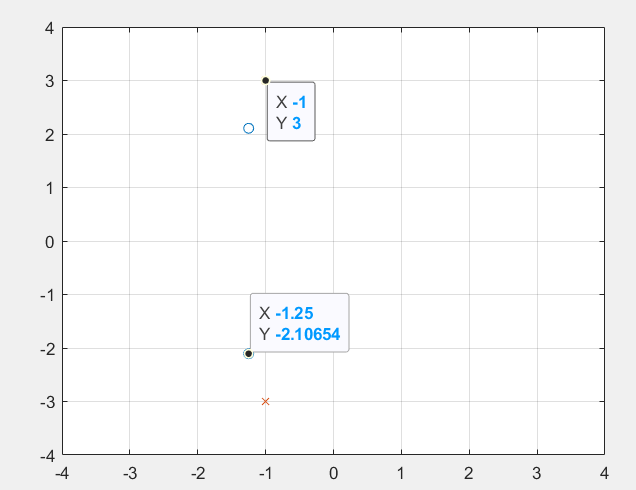
**problem 1：**

完整代码在problem1.m文件中。

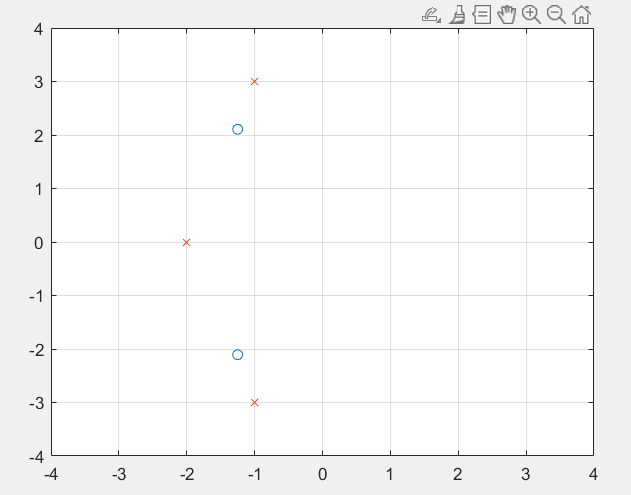
1. **Making Continuous-Time Pole-Zero Diagrams**
2. i.



ii.



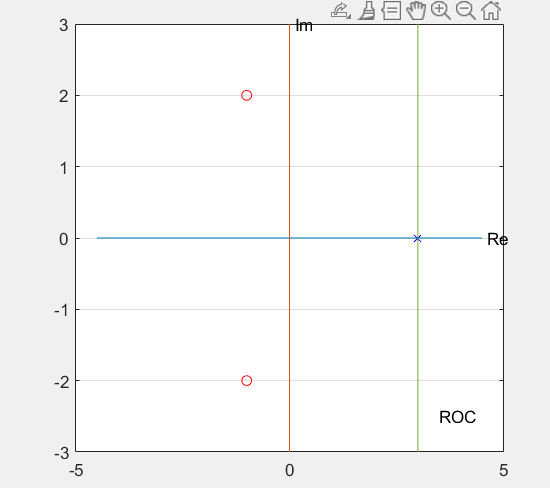
iii.



1. i. Re[pole1]=Re[pole2]=-1, so it has the ROC of Re[s]>-1

ii. Re[pole1]=Re[pole2]=-1, so it has the ROC of Re[s]>-1

iii. Re[pole1]=Re[pole2]=-1, Re[pole3]=-2, so it has the ROC of Re[s]>-1



1. pzplot能找到ROC的原理：

见pzplot.m中如下代码，

if nargin>2 % ROC optional

if(any(real(poles)<ROC)) % Any poles to the left?

lpole=max(real(poles(real(poles)<ROC)));

plot([lpole lpole],1.5\*[-MaxI MaxI],'-')

end

if(any(real(poles)>ROC)) % Any poles to the right?

rpole=min(real(poles(real(poles)>ROC)));

plot([rpole rpole],1.5\*[-MaxI MaxI],'-')

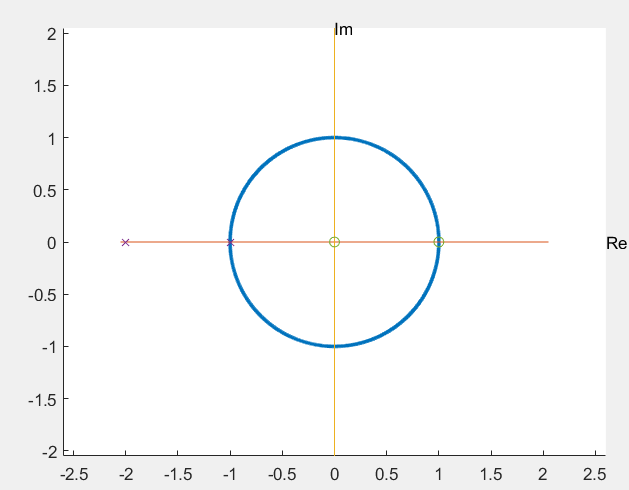
end

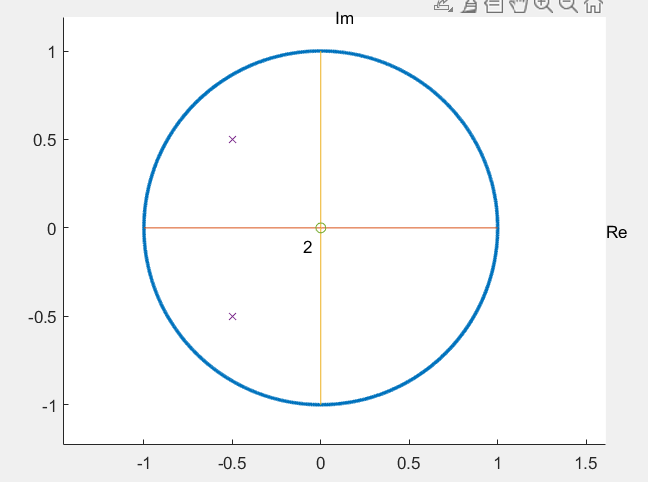
text(ROC,-1.25\*MaxI,'ROC') % Label the ROC

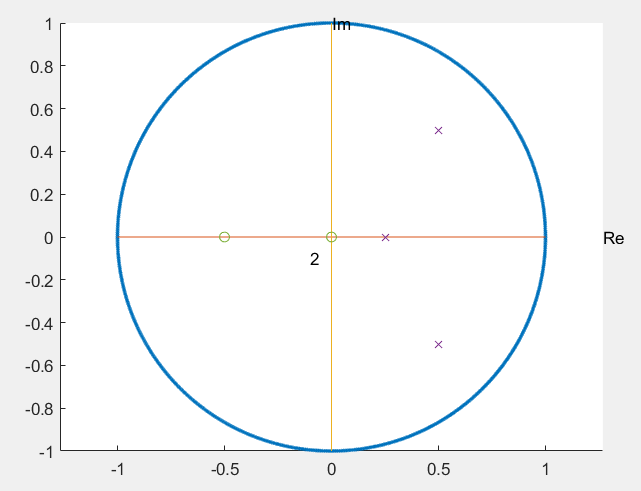
end

这段代码分别对ROC左侧的最大极点和右侧的最小极点，即距离ROC最近的两侧极点进行了寻找，而输入参数ROC的含义是系统函数的收敛域当中的某一点，则该点和左右的两条线自然形成了系统函数的收敛域。

1. **Making Discrete-Time Pole-Zero Diagrams**



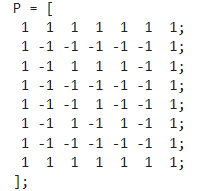




**problem 2:**

完整代码在problem2.m文件中。

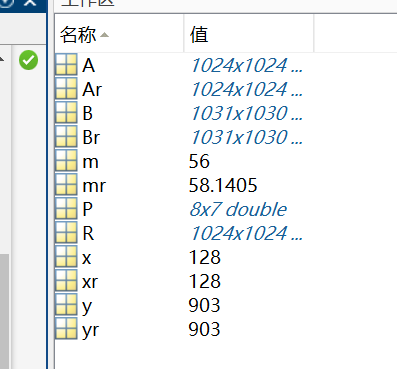
1. **smiley**
2. 当时，会在中间（）取到最大值。
3. 设计filter p为如下矩阵：



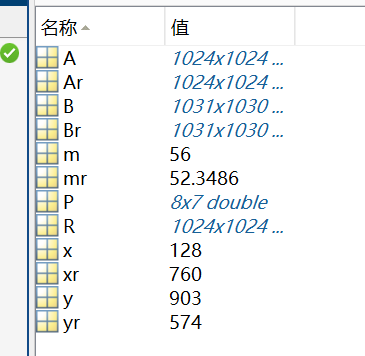
最后找到唯一点，反映在原图中：



1. 添加噪声，大部分情况filter工作良好。



加了噪声之后，也会有偶然情况导致需要找的图的值并非最大值，这取决于噪声幅度的相对大小，当其达到一定大小时，出现差错是无法避免的。

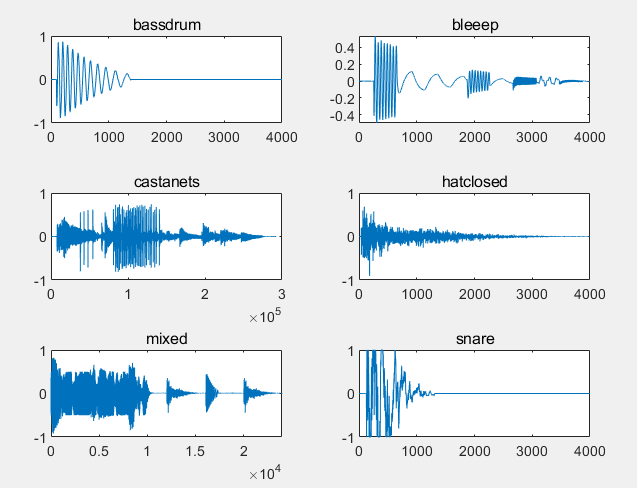


**problem 3:**

1. **Use the Sounds in Matlab**

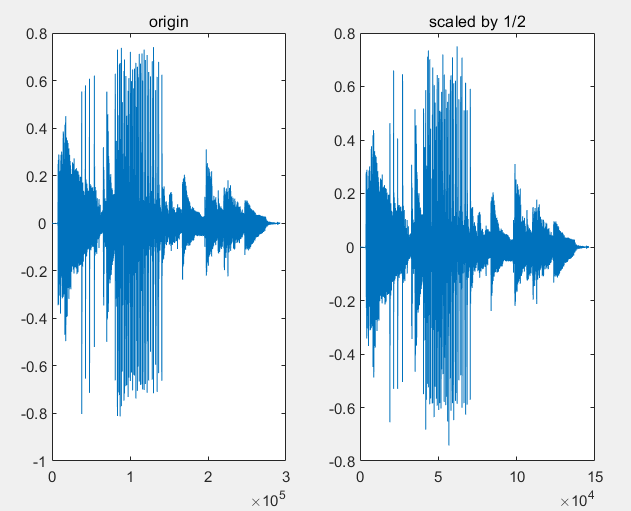
完整代码在problem31.m中。

**Exercise 1 & 2：**



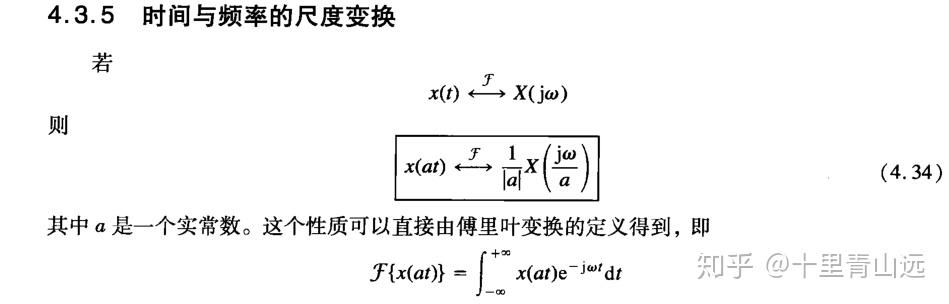
实际播放的感受与波形接近，幅度大的地方音量高。

**Exercise 3 & 4：**



倍速音频，会感觉到音调变高变尖；而减速音频，会感到音调减小；在音乐制作中，音调随频率改变而改变，当频率变大一倍时，音调正好高八度。

在奥本海姆的《信号与系统》连续时间傅立叶变换一章中关于傅里叶变换的性质介绍部分有如下内容：



相当于对连续信号傅立叶变换后所得表达式，除了一个的幅度因子外，信号在时间上也有一个线性尺度因子的变换，反之亦然。

按照上述公式，如果一段音频的放音速度比录制时的速度快，这就相当于上式中, 信号在时间上受到了压缩，傅立叶变换的频域就会拓展，此时听起来就会感到声音的频率变高了；反之，如果放音速度比原来慢，也就是, 时间延展，频域压缩，听起来频率就会降低。

**Exercise 5：**

修改fade.m的代码，出于应用考虑，level只能大于0、小于1，因此设定输入level为其他任何值时，默认为0。

switch(nargin)

case 2

if(level<=1&&level>=0)

t = linspace(1, level, length(x));

else

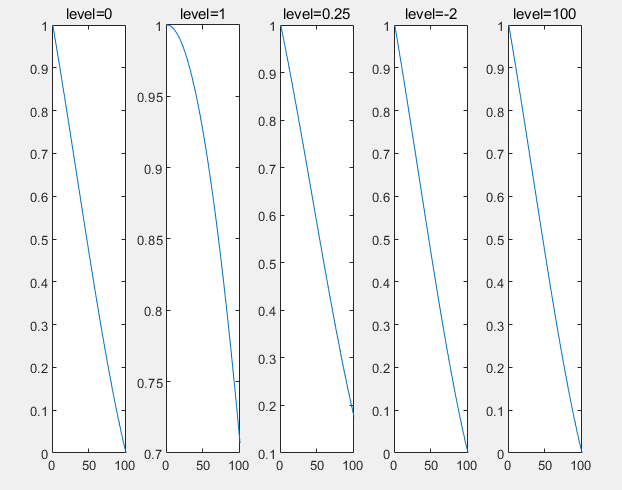
t = linspace(1,0,length(x));

end

case 1

t = linspace(1, 0, length(x));

end



**Exercise 6：**

编写repeat.m的代码，同样出于应用考虑，输入N<0时，默认为1倍。

t = sound;

switch(nargin)

case(1)

out = sound;

case(2)

if(N>0)

for i= 1:N-1

t = [t sound];

end

out = t;

else

out = sound;

end

end

运行以下代码，C1的长度是bassdrum的三倍，C2、C3则与bassdrum长度相当。

C1 = repeat(bassdrum,3);

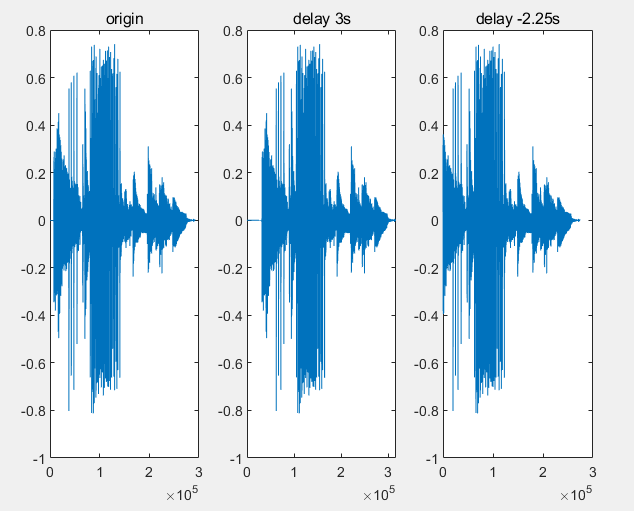
C2 = repeat(bassdrum,0);

C3 = repeat(bassdrum,-1);

% length of C1 = 12000, that of C2 and C3 = 4000.

**Exercise 7：**

编写delay.m代码，设定delay>0时，时延音频；delay<0时，跳过delay时间的音频。



**Exercise 8：**

参考soundsc函数编写mix.m：

zero = zeros(1,abs(length(sound1)-length(sound2)));

if(length(sound1)>length(sound2))

sound2 = [sound2 zero];

else

sound1 = [sound1 zero];

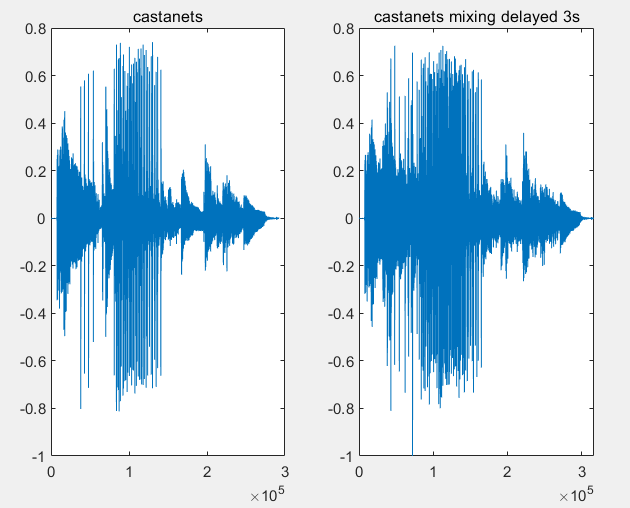
end

t = sound1 + sound2;

y = t / max(abs(t));

end

将castanets与自身延时3s后的结果相融，得到：

****

1. **Make Your Own Music**

详见problem32.m，使用函数gen\_wave()在gen\_wave.m中。