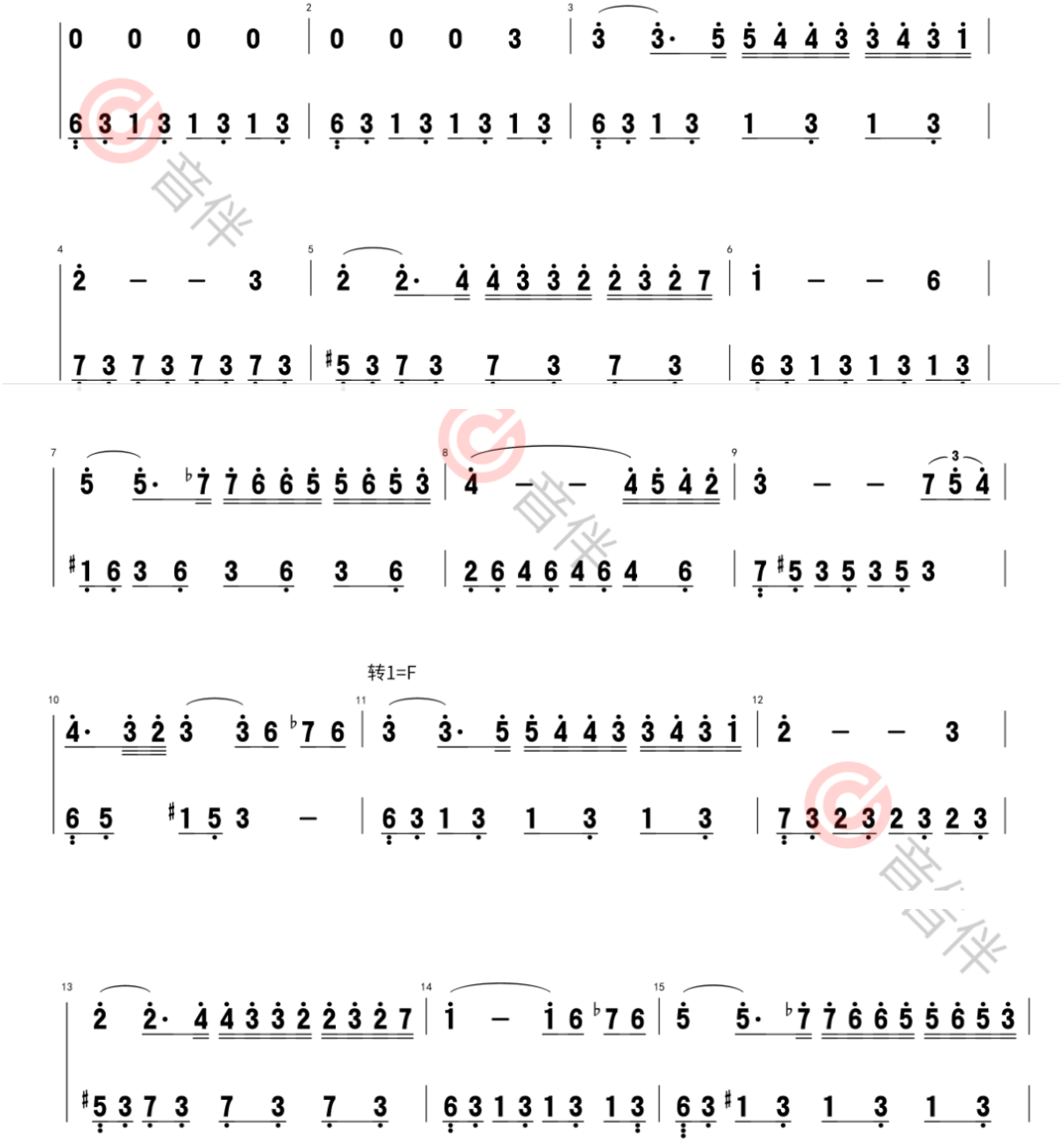
实验报告

一、实验目的

该实验旨在运用钢琴键的音频特征，结合信号处理技术，通过matlab拟合钢琴的声音并演奏《秋日私语》片段



二、实验步骤

1.调制钢琴音谱

在常见的钢琴上，第49个键，乐理中称为A4，作为调音标准音，现行的标准是440赫兹。其余键的频率满足以下公式：

$$f(n) = (2)^{\frac{n-49}{12}} * 440(Hz)$$

根据此规律，可以直接生成正弦的波并适当调整频率参数就可以得到钢琴音。弹奏代码如下：

```
%频率参数
freq = [130.81, 146.83, 164.81, 174.61, 196.00, 220.00, 246.94, ...  % 下八度C
        261.63, 293.66, 329.63, 349.23, 392.00, 440.00, 493.88, ...  % 中央C
        523.25, 587.33, 659.25, 698.46, 783.99, 880.00, 987.77, ...  % 上八度C
        1046.50, 1174.66, 1318.51, 1396.91, 1567.98, 1760.00, 1975.53,0];
%音符谱
notes = [6,10,15,10,15,10,15];
%采样率
Fs = 44100;
audio_left = [];
for i = 1:length(notes)
    f = freq(notes(i));
    duration = 0.4;
    t = 0:1/Fs:duration;
    y = sin(2 * pi * f * t);
    audio=[audio,y]
end
sound(audio, Fs);
```

2.加入节奏

钢琴中有各种各样的节奏，我们需要自定义的去设置每个音延长的时间duration来表示不同的节奏.

```
notes = [6,10,15,10,15,10,15];
Fs = 44100;
durations=[0.4,0.4,0.4,0.4,0.2,0.2,0.3]%自定义音符时长
audio_left = [];
for i = 1:length(notes)
    f = freq(notes(i));
    duration = duration(i);
    t = 0:1/Fs:duration;
    y = sin(2 * pi * f * t);
    audio=[audio,y]
end
sound(audio, Fs);
```

在演奏的片段中，以四分音符为一排每小节4拍，可以设定四分音符为0.8s，八分音符为0.4s，十六分音符为0.2s，三连音为0.3s，0.3s，0.2s，附点音符为0.6s和0.2s

3.两手联弹

我们用同样的方式生成左手音频，在这首曲子中，左手部分的力度较轻，所以我们缩放一下左手的振幅（该系数需要不断调整以找到最合适的）。在设置的过程中，需要注意左右手每小节的时间相同。同时，在信号相加时，左右手的audio会相差几个小的单位，我选择直接补0在首尾即可，毕竟在庞大的数据下这几个0并不会产生太大的影响。

```
%修正个别小偏差
audio_left=[audio_left,0,0,0,0,0,0];
audio_right=[0,0,0,0,0,0,0,audio_right,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
% 播放音频
sound(0.012*audio_left+audio_right, Fs);
```

4.爆破音处理

运行上面的代码，会发现在每个音符开始前会有很大的爆破音。这是因为当音符跨度大，不同音频间振幅跨度大，而且持续时间短，听感上导致爆破音的出现。

一种方法是渐入渐出的技术处理运用了插值技术，在音符变换之间添加一些点并作插值，使得音的变换更加平缓。另外一种方法就是给音的首尾乘上卷积核进行平滑化处理函数。

法一：线性插值

```
for i = 1:length(notes)
    f = freq(notes(i));
    duration = lengths(i);
    t = 0:1/Fs:duration;
    y = sin(2 * pi * f * t);
    % 应用渐入渐出
    fade_duration = 0.2;
    fade_samples = round(fade_duration * Fs);
    fade_in = linspace(0, 1, fade_samples);
    fade_out = linspace(1, 0, fade_samples);
    y(1:fade_samples) = y(1:fade_samples) .* fade_in;
    y(end-fade_samples+1:end) = y(end-fade_samples+1:end) .* fade_out;
    audio = [audio, y];
end
```

法2：高斯卷积

```
% 定义高斯核的标准差和长度
sigma = 0.005;
kernel_size = round(6 * sigma * Fs);
if mod(kernel_size, 2) == 0
    kernel_size = kernel_size + 1;
end

% 生成高斯核
t = linspace(-3*sigma, 3*sigma, kernel_size);
gaussian_kernel = exp(-t.^2 / (2 * sigma^2));
gaussian_kernel = gaussian_kernel / sum(gaussian_kernel);

% 对音频信号进行高斯卷积
smoothed_audio = conv(audio, gaussian_kernel, 'same');
m1=max(smoothed_audio);
m2=max(audio);
smoothed_audio=smoothed_audio/m1*m2*4;
% 播放平滑后的音频
sound(smoothed_audio, Fs);
```

视频1为未处理爆破音的音乐：可以明显听到想打鼓一样的爆破声，视频2为处理过爆破音的音乐，明显平滑很多

结果1为考虑前四个步骤的成果，此时音色与钢琴还有比较大的差别。

5.基音与泛音

钢琴声音是由基音和多个泛音组成的。基音是音符的主要频率，而泛音是基音频率的整数倍，它们共同决定了钢琴音色的丰富性和独特性。当基音振动时泛音同时振动，但需要调整不同频率前的系数以更好地模拟钢琴声音的特性。这个系数的调整非常重要，每个音都有区别。本模型中简化处理，低音区我用5个音一调，其音色比较接近。高音区3个或2个一调，其音色差别较大。

下面是调试中的一个代码模板

```
clear, clc;
%定义音符对应的频率
freq = [130.81, 146.83, 164.81, 174.61, 196.00, 220.00, 246.94, ...  % 下八度C
    261.63, 293.66, 329.63, 349.23, 392.00, 440.00, 493.88, ...  % 中央C
    523.25, 587.33, 659.25, 698.46, 783.99, 880.00, 987.77, ...  % 上八度C
    1046.50, 1174.66, 1318.51, 1396.91, 1567.98, 1760.00, 1975.53,0];

%定义每个音符
notes_left = [1,2,3,4,5];
Fs = 44100;
% 泛音相对基音的幅度（权重）
harmonicAmplitudes = [1, 0.5, 0.6];  % 基音和泛音
audio_left = [];
for i = 1:length(notes_left)
    f = freq(notes_left(i));
    duration = 0.5;  % 音符总时长
    t = 0:1/Fs:duration;

    % 生成基音和泛音
    y = zeros(size(t));
    for k = 1:length(harmonicAmplitudes)
        y = y + harmonicAmplitudes(k) * sin(2 * pi * f * k * t);
    end

    audio_left = [audio_left, y];
end
sound(0.1*audio_left, Fs);
```

视频3为未加泛音，视频4为加了泛音的结果，很明显4的音色更像钢琴

6.ADSR包络

ADSR包络是描述声音强度随时间变化的模型。它常用于合成器和数字音频处理，以模拟真实乐器的动态特性。对于钢琴音符，ADSR包络可以大致描述如下：

1. **Attack**：音符被按下时声音迅速达到最大音量的时间，钢琴非常快。
2. **Decay**：声音从最大音量下降到持续音量的时间。此首取词有踏板，下降时间较长。
3. **Sustain**：音符被按住时的稳定音量。钢琴稳定

4. **Release**：音符被松开后声音逐渐消失的时间。此首曲子有踏板，消失时间较长

实现的思路就是根据不同阶段对振幅进行适当的缩放以模拟音色：

```
% 钢琴的ADSR包络参数
attackTime = 0.01;  % 攻击时间
decayTime = 0.2;    % 衰减时间
sustainLevel = 0.4; % 持续级别
releaseTime = 0.3;  % 释放时间
%循环体内

    % 生成ADSR包络
    env = zeros(size(t));
    attackSamples = round(attackTime * Fs);
    decaySamples = round(decayTime * Fs);
    releaseSamples = round(releaseTime * Fs);
    sustainSamples = length(t) - (attackSamples + decaySamples +
releaseSamples);

    env(1:attackSamples) = linspace(0, 1, attackSamples);
    env(attackSamples+1:attackSamples+decaySamples) = linspace(1,
sustainLevel, decaySamples);

 env(attackSamples+decaySamples+1:attackSamples+decaySamples+sustainSamples)
= sustainLevel;
    env(end-releaseSamples+1:end) = linspace(sustainLevel, 0,
releaseSamples);
    % 应用ADSR包络
    y = y .* env;
```

视频5为处理过的声音，视频6为未处理的声音，明显视频5的音色更接近钢琴

三、实验结果

结果2为模型最终运行结果