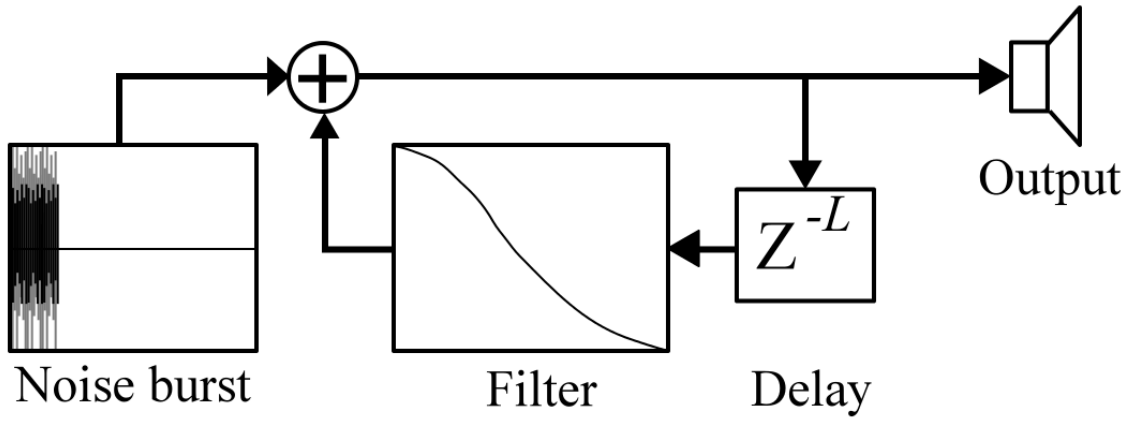
合成吉他音效

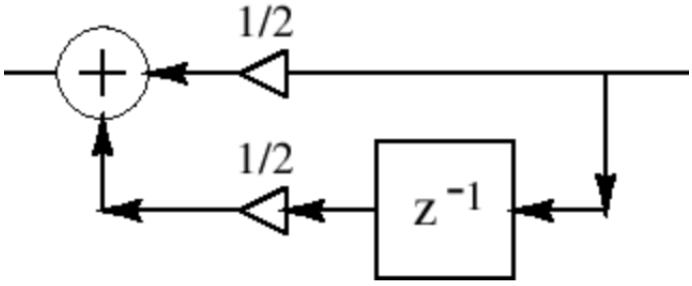
董峦，胡春华　新疆农业大学　2018

实验背景

Karplus-Strong 算法（Karplus Strong Algorithm）是一种声音合成方法。该算法如此命名是因为 Alexander Strong 发明了这个算法，而 Kevin Karplus 首次分析了它的工作原理。他们一起开发了相应的软硬件，并将该算法命名为 "Digitar"，即 "digital guitar" 之意。该算法工作原理如下图所示



图中‘Noise burst’是输入，一般是高斯白噪声（均值为0，方差为1的随机信号），可以看到信号处理过程中存在反馈，即信号被延迟L个样点后通过一个低通滤波器再叠加到信号上去。滤波器的增益应小于1。在最初提出的算法中，滤波器的运算是对两个相邻样点取平均，如下图所示



根据上面的描述，我们可以写出Karplus-Strong 算法的差分方程是：

$$y_i = \frac{1}{2}(y_{i-L} + y_{i-L-1}) + x_i \qquad (1)$$

下面我们将根据该差分方程合成一个声音信号y，并且用卷积计算来实现，接着从频域理解该算法。关于该算法的更多信息可参考维基百科页面Karplus–Strong_string_synthesis

实验内容

首先我们可以观察（听和看）一下该算法产生的信号，运行如下代码

```
% 调用 karplus_strong 函数，听听合成出的声音有没有吉他的音色
x = randn(1, 100);     % x 是正态分布随机数，在这里指输入的白噪声信号
N = 48000;             % 输出样点数
L = 50;                % 延迟
y = karplus_strong(x, L, N); % 产生输出
Fs = 48000;                  % 采样率48KHz，在这里该数值可人为指定
soundsc(y, Fs);              % 将信号当作声音来播放
plot(y);                     % 观察信号
```

在上述代码中调用了已经编写好的 karplus_strong 函数，该函数实现了上述描述该算法的差分方程。下面我们需要用卷积运算来得到输出信号y，以检验线性移不变系统的性质，即齐次性和叠加性，运用这两个性质我们推导出这类系统的响应 $y = x * h$ 其中x是系统的输入信号，h是系统的单位冲激响应，它们之间的运算是卷积。也就是说线性移不变系统可以用其单位冲激响应来描述。

那么怎样得到Karplus-Strong系统的单位冲激响应呢，其实从单位冲激响应的定义出发就可以了，即给系统输入单位冲激信号，系统的输出就是单位冲激响应。下面的代码能够得到该系统的单位冲激响应 h

```
% 获得系统单位冲激响应
h = karplus_strong(ones(1), L, N);  % 用长度为1的向量表示单位冲激信号
stem(h(1:400));                     % 观察单位冲激响应
```

下面我们需要通过卷积运算 $x * h$ 得到输出信号，我们这样定义本次实验的卷积运算

$$y_i = \sum_{k=1}^{n} x(i+1-k)h(k) \tag{2}$$

之所以 x 的元素下标是 $(i+1-k)$ 是因为MATLAB向量元素下标是从1开始。根据上式完成 karplus_strong_conv 函数的编写，即从定义出发用卷积运算根据 x 和 h 得出 y。编写完该函数后，用下列代码检验代码的正确性。

```
% 检验计算结果与实际结果的差异
y2 = karplus_strong_conv(x,h);
assert(sum(abs(y - y2)) < 1e-6, '卷积计算结果与实际结果差异过大');
```

你也可以用MATLAB自带的卷积计算函数检验实现的正确与否，同时比较一下程序的计算效率如何

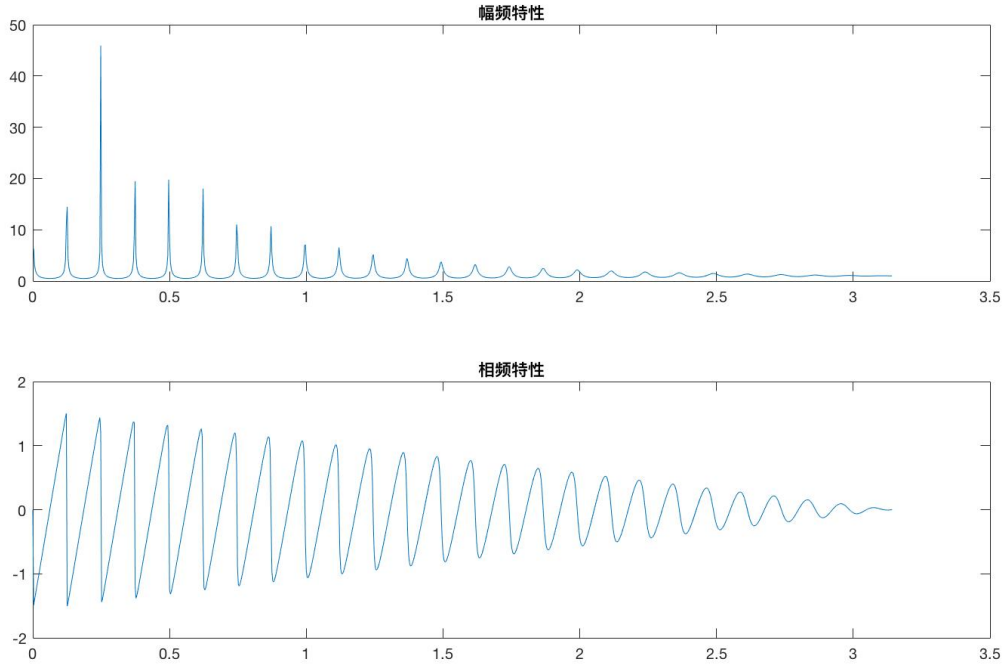
```
y3 = conv(x, h);
assert(sum(abs(y2 - y3(1:length(h)))) < 1e-6, 'karplus_strong_conv 函数实现有误');
```

下面我们对该系统进行频域分析。我们对差分方程进行Z变换后能够得到系统的传递函数

$$H(z) = \frac{1}{1 - 0.5(z^{-L} + z^{-L-1})} \tag{3}$$

令 $z = e^{j\Omega}$ 就是系统频率响应。运行下列代码，观察该系统的频率特性

```
omega = linspace(0, pi, 1000);  % omega 即归一化频率，令其在0到pi上取值
H = 1 ./ (1 - 0.5*(exp(-1j * L * omega) + exp(-1j * (L+1) * omega)));  % 计算频率响应
subplot(211)
plot(omega, abs(H));    % 画幅频特性
title('幅频特性');
subplot(212)
plot(omega, angle(H));  % 画相频特性
title('相频特性');
```

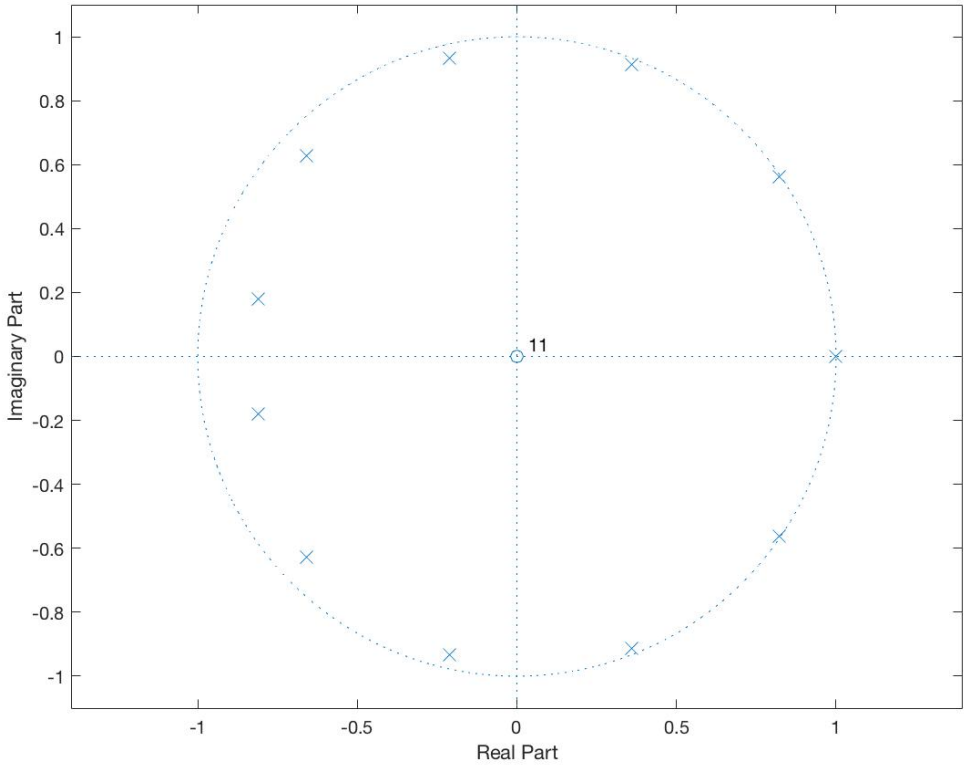


从幅频特性上可以看到，在几个均匀分布的频率位置，信号得到了增强，而其他频段的信号被抑制了，也就是说谐波得到了增强，这与乐音的原理有几分相似。我们知道白噪声的频谱含有所有频率成分，根据上面展示的幅频特性，你能从频域解释该声音合成算法的工作原理吗?

系统的这个特性在零极点图上也能反映出来。

```
% 零极点分析
% 以 L=10 为例
% 此时 H(z) = z^11 / (z^11 - 0.5z - 0.5)
% 即 H 在原点有11个零点
zero_points = [0 0 0 0 0 0 0 0 0 0 0]';

% 还有11个极点可以这样确定
% 利用 roots 函数求多项式方程：z^11 - 0.5z - 0.5 = 0 的根
% roots 里的向量是多项式：z^11 - 0.5z - 0.5 的向量化表示
pole_points = roots([1 0 0 0 0 0 0 0 0 0 -0.5 -0.5]);
% 画零极点图
figure();
zplane(zero_points, pole_points);
```



该零极点图上，极点与单位圆很近且均匀分布，这便解释了幅频特性的形态。试问，极点的分布与哪个参数有关呢?

小结

本次实验我们探讨了一个声音合成算法的原理。合成声音是一个信号处理过程，我们从差分方程出发可以计算出输出，利用输入与单位冲激响应的卷积也能计算。

紧接着我们通过观察系统的频率特性和零极点图能够进一步理解系统的工作过程，即系统在频域是如何加工信号的。以上展示的分析过程可以为你所用，分析其它你感兴趣的数字信号处理系统。

输入噪声 x 的长度和延迟参数L会影响合成声音的音调，第一个框图中的滤波器也可以，但本次实验限定了滤波器的实现为两个相邻样点的平均，所以你只能更改 x 的长度和参数L去观察不同的配置如何影响合成效果。重新合成一个音符听听吧。