プロジェクト B　非線型処理に基づく歪んだ音色への加工　2012/12/16

ポピュラー音楽のエレキギターパートで多用されるエフェクターに「ディストーション」や「オーバードライブ」と呼ばれるエフェクターがある。これらのエフェクターは、ある種の電気回路が非線型な増幅を行なうことを積極的に利用したものである。これらのエフェクターが行う音色の加工を MATLAB で再現する方法を考えてみる。

1　周期関数

音程のある楽器の音は、周期性がその基本となっている。周期関数の性質について簡単に整理する。
周期関数は次の式を満たす関数である。

$$x(t) = x(t+T) \tag{1}$$

ただし、T は周期である。

この関数となるような信号を MATLAB で簡単に作成するには、1 周期のデータを必要な長さの分だけ繰り返せばよい。

```
>> Fs=8000;
>> r=-1+2*rand(64,1);
>> rr=repmat(r,125,1);
>> soundsc(rr,Fs)
>> plot(log(abs(fft(rr(1:512).*hann(512)))))
```

スペクトルを観察すればわかるが、1 周期のデータがランダムであっても、高調波構造 (倍音を持つ構造) となる。周期の周波数が基本周波数となる。

このような仕組みを応用して 1 周期分のデータを保持し、それを繰り返すことで楽器音を生成するのがシンセサイザーの wavetable 合成方式である。

練習 1 適当な音を 1 周期分用意し、それを繰り返して 約 1 秒の音を生成してみよ。

2　非線型処理

単純な仕組みで歪ませるエフェクターにファズがある。MATLAB で信号 `s` をファズのように歪ませるには、次のようにすればよい。

```
>> sound(s*5,fs)
```

ただし、`fs` は `s` のサンプリング周波数だとする。音が小さい場合は、`5` の部分をより大きな値に変更すると歪むだろう。

これは、`sound` 関数では、`help` で「`Y` の値は、範囲 `-1.0 < = y <= 1.0` と仮定します。この範囲外の値は切り取られます。」と書かれているように、値が切り取られることで波形が変化することで生じる。

このことを入力と出力の関係で見てみる。上記の処理は、入力が 5 倍され、出力の絶対値が 1 以上になったらそれ以上絶対値を大きくしない処理である。入力を x 、出力を y としてその関係を式で表すと次のようになる。

$$y = \begin{cases} 1 & (y \geq 1) \\ -1 & (y \leq -1) \\ 5x & (otherwise) \end{cases} \tag{2}$$

この式を関数にしてみる。

```
function output = fuzz_trans( inputSignal )

output=zeros(size(inputSignal));
for iSignal = 1:length(inputSignal),
    s=inputSignal(iSignal);
    signSignal = sign(s);
    s=abs(s);
    if s >= 1/5,
        output(iSignal)=signSignal;
    else
        output(iSignal)=signSignal*(5*s);
    end
end
end
```

この関数を利用すると歪んだ信号を生成できる。

```
>> plot(fuzz_trans(g)); ylim([-1.5 1.5])
```

この関数の入出力の関係をプロットしてみる。

```
>> plot(-1:1/100:1,fuzz_trans(-1:1/100:1)); ylim([-1.5 1.5])
```

単に、入力を 5 倍増幅して出力する場合には、$y = 5x$ となり、入出力の関係のプロットは直線となる。しかし、上記のプロットでは、直線とならない。このように、直線とならない関係を非線型な関係とよぶ。

`fuzz_trans` の効果を観察するために、正弦波を処理してみる。

```
>> Fs=8000;
>> t=0:1/Fs:1;
>> s440=sin(2*pi*440*t);
>> f440=fuzz_trans(s440);
>> subplot(2,1,1); plot((0:49)/fs,f440(1:50)); ylim([-1.5 1.5])
>> subplot(2,1,2); plot((0:49)/fs,s440(1:50)); ylim([-1.5 1.5])
```

このように、非線型な処理をしても周期が保たれていることはわかる。

練習 2 この観察結果から、`fuzz_trans` は、入力の音の ＿(1)＿ を変化させるが、 ＿(2)＿ は変化させないことがわかる。

下線部 (1), (2) に入る語句を考えよ。

処理結果のスペクトルを観察する。

```
>> plot((0:511)*Fs/512,log(abs(fft(f440(1:512).*hann(512)'))))
```

元の正弦波の成分だけでなく、3,5,7 と奇数倍の倍音成分が加わっていることがわかる。このように、非線型処理では、入力には存在していなかった倍音成分が新たに加わる。この点が、線型フィルタリングなどの処理とは大きく異なる。

練習 3 440Hz 以外の周波数の正弦波に `fuzz_trans` を適用するとどのようなスペクトルになるか推測せよ。また、実際にスペクトルを観察してみよ。

3 非線型処理を利用したプログラミング

古いギターアンプは増幅のための素子として真空管を利用していた。真空管は、入出力の関係が非線型 (曲線) になるため、大きな音にすると歪みが目立つ。

曲線的な入出力関係を実現する関数として例えば次のような関数が考えられる。

```
function output = nonlinear_test(inputSignal)

output=zeros(size(inputSignal));
for iSignal = 1:length(inputSignal),
    s=inputSignal(iSignal);
    signSignal = sign(s);
    s=abs(s);
    if s > 2/3,
        output(iSignal)=signSignal;
    else
        output(iSignal)=signSignal*(-9/4*s^2+3*s);
    end
end
end
```

適当な入力を、絶対値の最大値が 1 になるように正規化し、その信号を `nonlinear_test` で処理すると、歪んだ音色が得られる。

つまり、処理に利用する関数を適切に設計できれば、望むような音色を得ることができる。

また、存在するエフェクターに関しては、適当な正弦波を入力することにより歪み方を推測することができる。また、その歪み方を実現するような関数を推定できれば MATLAB で実現できることがわかる。