4　応用問題 (2): ピッチシフターを作る

4.1　概要

配布資料にあるように、ピッチシフターは、入力のピッチ (音程) を変化させるエフェクターである。

このエフェクターでは、音声データを間引いて元のサンプリング周波数で再生すると、高く聞こえたり、サンプリング周波数を半分に設定して再生すると、低く聞こえる原理を利用している。

ただ、そのままでは、高くすると、再生される長さが短くなってしまう (例えば、倍の高さにする、つまり 1 オクターブ上げると長さが半分になってしまう) し、低くしようとすると、長さが長くなってしまう。

そこで、ピッチを上げる場合には、ピッチを保ったまま時間を引き延ばし、下げる場合には時間を短縮しなければならない。まず、この時間の伸縮をおこなう関数を作成する。

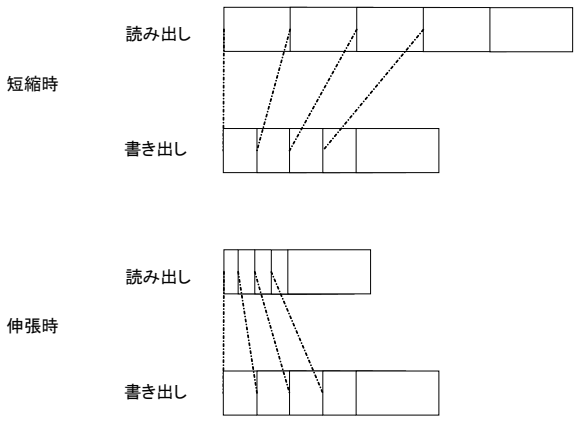
4.2　タイムシフト

ピッチを保ったまま時間の伸縮をする方法はいくつも提案されているが、ここでは Real Sound Synthesis for Interactive Applications (Cook, 2002) に紹介されている方法を用いる。

この方法は、人間がピッチ (音の周期性) を感じるのは、20ms 以上の単位であることを利用する。ピッチを感じる適当な単位を重ねたり繰り返したりすることで伸縮する。

手順は以下の通りである。

1. 入力の読み出し時刻、出力の書き出し時刻を初期化する。
2. 読み出し時刻から 180ms の区間を取り出す。
3. 取り出した区間を書き出し時刻から足し合わせる。
4. 書き出し時刻に 80ms を加える (固定長とする)。
5. 読み出し時刻を更新する。80ms 加えると、伸縮しない。伸び縮みさせるためには、この読み出し時刻を適当に変化させればよい。
6. 2 から 5 を繰り返す。

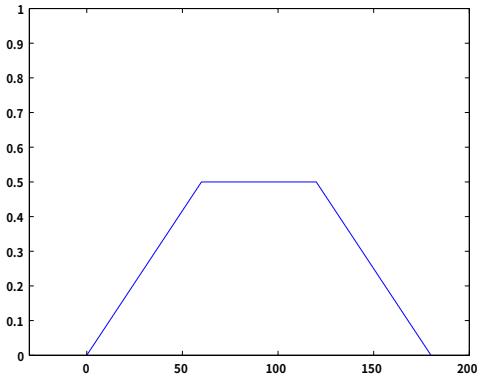


課題 1　下記の関数 `timeshift` を完成させよ。

```
function wave = timeshift(input, fs, factor)
bsize=0.08;
outstep=_______;
instep=floor(outstep/factor);
```

```
frame=floor(outstep*0.75);
t1=1:frame;
win(t1)=_______;
win(frame+t1)=_______;
win(frame*2+t1)=_______;
inbase=0;
outbase=0;
wave=zeros(floor((length(input)-3*frame)/instep)*outstep+3*frame,1);
i = 1:3*frame;
for j=1:ceil((length(input)-3*frame)/instep),
    wave(outbase+i)=_______;
    inbase = inbase + instep;
    outbase = outbase + outstep;
end
```

ただし、関数中の `win` は下図のような時間窓である。



4.3 `pitchshifter` 関数の作成とその応用

課題 2 下記の関数 `pitchshifter` を完成させよ。

```
function wave = pitchshifter(input, fs, factor, len)
if (nargin < 3) factor = 1; end;
if (nargin < 4) len = 1; end;
[n, d] = rat(factor);
r=resample(_________________);
wave=_________________;
```

課題 3 適当なピッチのある音 (楽器音とか母音) を (できるだけ高いサンプリング周波数で適当な長さ) 録音し、`pitchshifter` 関数で様々な条件で加工せよ。

課題 4 様々な加工結果の波形やスペクトルを観察せよ。その結果を元に、今回使用したアルゴリズムの特徴、限界などを考察せよ。また、それらの限界を改善するには、どのような手法があるのか、調べてみよ。

課題 5(おまけ) 適当なピッチのある音 (楽器音とか母音) を (できるだけ高いサンプリング周波数で適当な長さ) 録音し、その音の高さを自己相関関数を用いて推定せよ。

課題 6(おまけ) そのデータを元に、`pitchshifter` 関数を使えば、必要な長さ、音階の音を作成することができる。これを `note` 関数の代わりに用いれば、自分で録音した音で曲を演奏することができる。このスクリプトを作成せよ。