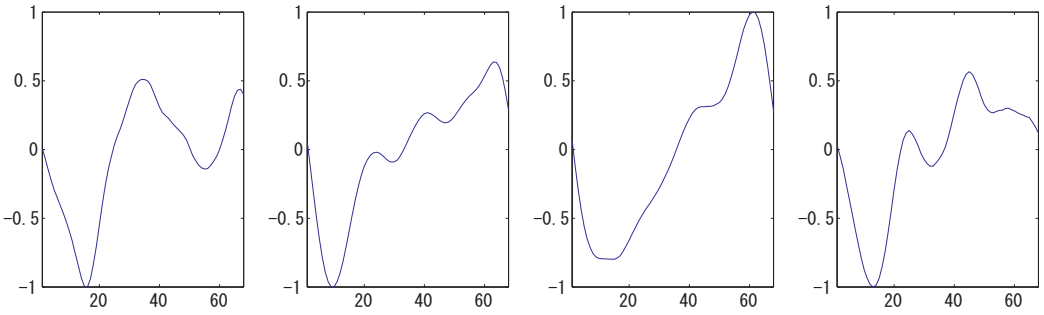
4　複数の波形データを用いた合成

2 章でも述べたように、実際の楽器の音は、時間によって音色が変化する。とりあえず、プログラムを簡単にできるように ADSR の区間に対応させて変化させることを考える。

4 つの区間から 3 章の手法で、それぞれ 1 周期分を抽出したものをプロットすると次のようになる。



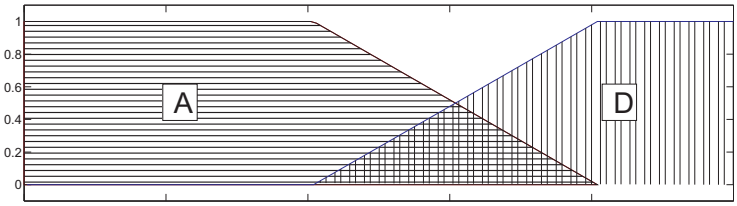
区間によって波形が異なることがわかる。

それぞれの区間のデータを `a,d,s,r` に格納しているとすると、次のスクリプトで上記のようにプロットできる。

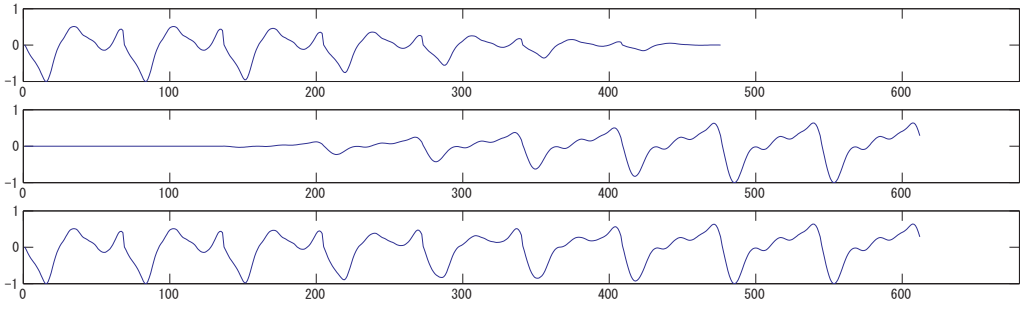
```
>> wavetable(:,1)=a;
>> wavetable(:,2)=d;
>> wavetable(:,3)=s;
>> wavetable(:,4)=r;
>> len=length(wavetable(:,1));
>> for i=1:4,
subplot(1,4,i); plot(------------------); xlim([1 len]); ylim(--------);
end
```

これらの波形は、同じ周期を持っている。しかし、例えば、A から D に遷移する場合に、直接異なる波形をつなごうとするとなめらかにつながらない場合もある。

そのような場合に、なるべくなめらかにつなぐ手法として、二つの波形を徐々に比率を変化させて足し合わせる手法がある。下図のように、重なりあうところでは、先行する波形 (ここでは A) を直線的に減少させ、後続の波形 (ここでは D) を直線的に増大させる。



重なる部分を 5 周期分の長さだとすると実際には、次のようになる。(上段が A の波形、中段が D の波形、下段は足し合わせた波形である)



ここまでの考え方に基づき、ADSR エンベロープにあわせて、音源を切り替えて音を合成する MATLAB プログラムを考える。

まず、ADSR エンベロープを作成する関数を作成する。

```
function env = ADSR2(format, fs)
%ADSR2 ＡＤＳＲエンベロープで振幅変調を行う窓関数を作成
% format は以下の行列
% [attack_time(秒) attack_level(1.0を最大とする相対値);
%  decay_time(秒) decay_level(1.0を最大とする相対値);
%  sustain_time(秒) sustain_level(1.0を最大とする相対値);
%  release_time(秒) 0]
% 音符の長さに合わせてこのフォーマットを決める。
% それぞれの時間は継続時間長にする。
% attack, decay の時間は余り変えない方が多分よいので、
% 上位の関数を作成した方がよい
% fs はサンプリング周波数(Hz)
dur = sum(_________); % ADSR エンベロープの全体の長さ(秒)を計算
len = length(0:1/fs:dur);
env = zeros(__________); % 全体の長さ分のベクトルを予め作成
fin = 1;
amp = 0;
for i = _________,
    start = fin;
    range = floor(__________________);
    fin = start + range;
    env(start:fin) = ______ + (________ - ____) * (0:1/________:1);
    amp = _________________;
end
```

この ADSR2 を使って音を生成するプログラムは次のようになる。table には上記の wavetable のように、ADSR に対応したデータが配列として格納されていることを想定する。

```
function wave = wavetable_synth1(table, format, fs)
%WAVETABLE_SYNTH1 wavetable 合成法の簡単な実装
% table に格納されたデータのピッチ通りに
% ADSR エンベロープで振幅を変化させて合成する
% format は以下の行列
% [attack_time(秒) attack_level(1.0を最大とする相対値);
%  decay_time(秒) decay_level(1.0を最大とする相対値);
%  sustain_time(秒) sustain_level(1.0を最大とする相対値);
%  release_time(秒) 0]
tau = length(table); % 音源の周期を基本周期とする
ovlp = 5; % 隣り合う音源が重なる部分の長さ(何周期になるか)
wsize = _________________ ; % 重なる部分の点数
t = ___________________ ; % 重なる部分の時間軸の作成(縦ベクトルにする)
i_win = t/_____________ ; % 重なる部分で増大する窓関数
d_win = _____________ ; % 重なる部分で減少する窓関数
dur = ____________ ;
len = dur * fs; % 全体の長さ(点数)
pts = floor(format(:,1)*fs/tau); % ADSR のそれぞれの部分の点数
pts = pts - [1; repmat(2,size(format,1)-2,1); 1] * (ovlp - 1) / 2; % 重ね合わせの部分を除外
wave = [];
for i = _________________,
    wave = [_______; repmat(table(______),____________,1);
            repmat(________,______,1).*_____+repmat(________,_______,1).*i_win];
end
wave = [______; repmat(table(:,4),_______,1)];
if length(wave)>len, % 長さの調整
    wave = wave(1:len+1);
else
    wave(len+1) = 0; % 短い場合は足りない部分を 0 で埋める
end
wave = wave .* ADSR2(format,fs)';
end
```

この合成方法では、table に格納する音によって合成される音の質が変化する。また、区間の数を増やすことで、より細かい音色の変化を表現できる。