

# 【スペクトル包絡 LPC Spectral Envelope】

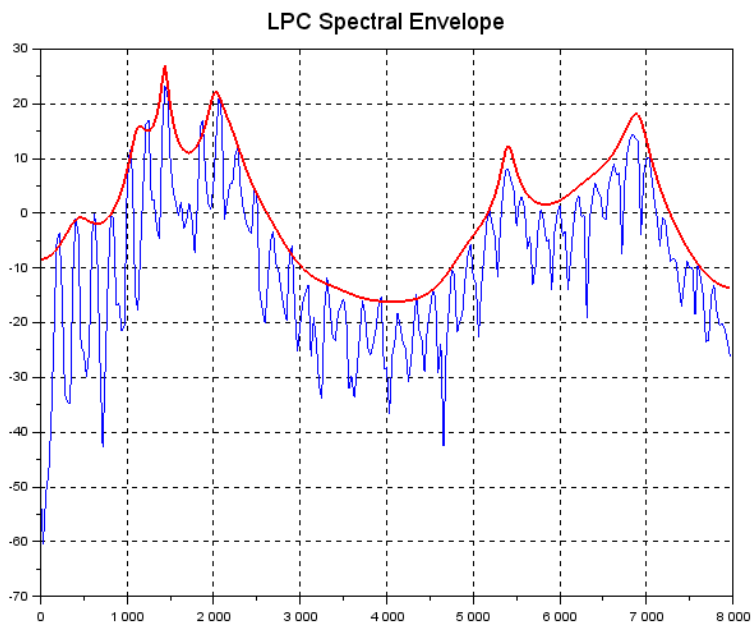


Figure 1: Scilab 実行結果

Source Code 1: Scilab

```
////////////////////////////////////  
//   スペクトル包絡  
//   Spectral Envelope  
//  
//           M. Tsutsui  
////////////////////////////////////  
  
clear;  
  
funcprot(0); // 自己相関関数  
function[res]=auto_corr(lag,signal); // lag: ラグ signal: 対象信号  
  
    size=length(signal);  
  
    loop=0;  
    res=[];  
  
    for k=0:1:lag+1; // ラグ 0~lag+1 まで  
        for n=1:1:size-k;  
            if(n+k>size)  
                signal(n+k)=0;  
            end  
            loop=loop+signal(n)*signal(n+k);  
        end  
  
        res=[res,loop];  
  
        loop=0; // 値初期化  
    end  
  
endfunction
```

```

funcprot(0); //レビンソン・ダービンアルゴリズム関数
function[alpha]=my_lev_durb(order,rxx); //引数 order:次数 r:相関係数 auto_corr関数に関して,次数+1個必要
//-----初期値-----//
alpha=[]; //LPC係数
alpha_d=[]'; //LPC係数一時保持用
alpha_d(1)=1;
k=[]; //PARCOR係数
k(1)=0;

w=0;
u=[];

alpha(1)=1; //LPC係数 固定
alpha(2)=0; //LPC係数初期化

w(1)=rxx(2); //r(1)
u(1)=rxx(1); //r(0)

//-----m=1__must処理-----//
k(1)=0;

k(2)=w(1)/u(1);
u(2)=u(1)*(1-k(2)^2);

alpha(2)=alpha(2)-k(2)*alpha(1);

alpha_d(2)=alpha(2);

w(2)=alpha(1)*rxx(3)+alpha(2)*rxx(2); //i=2

alpha(1)=1;
alpha(3)=0;

//-----m=2以上処理-----//
for m=2:1:order+1;

    k(m+1)=w(m)/u(m);
    u(m+1)=u(m)*(1-k(m+1)^2);

    alpha(2)=alpha(2)-k(m+1)*alpha(m);

    alpha_d_buf=[];
    for o=1:1:m-1;
        alpha_d_buf=[alpha_d_buf,alpha_d(o)];
    end

    alpha_d_buf=flipdim(alpha_d_buf,2); //flipdim:配列反転

    for i=1:1:m-1;
        alpha(i+2)=alpha(i+2)-k(m+1)*alpha_d_buf(i);
    end

    for j=2:1:m+1;
        alpha_d(j)=alpha(j);
    end

    alpha(1)=1;
    alpha(m+2)=0;

    alpha_buf=[];
    for r=1:1:m+1;
        alpha_buf=[alpha_buf,alpha(r)];
    end
end

```

```

        rxx_buf=[];
        for h=2:1:m+2;
            rxx_buf=[rxx_buf,rxx(h)];
        end

        w(m+1)=sum(alpha_buf.*flipdim(rxx_buf,2));

    end

    alpha=alpha(1:(length(alpha)-1));
endfunction

s_size=512;//サンプルサイズ

fs=8000;//サンプリング周波数

wav_data=wavread('LPC_test.wav');//wavファイル読み込み

order=50;//次数

data=wav_data(1:s_size);

//-----プリエンファシス処理-----//
data_pre=[];
data_pre(1)=0;
for i=2:s_size;
    data_pre(i)=data(i)-0.98*data(i-1);
end

//-----窓処理-----//
data=data_pre.*window('hn',s_size);//ハニング窓を乗算

data_fft=fft(data);

spectrum=20*log10(abs(data_fft));//信号 スペクトル

f=0:s_size/2-1;

x_scaling=f*fs/s_size;//横軸 スケーリング

r=auto_corr(order+1,data);//相関係数 自作レビンソン・ダービン関数使用より,order+1個の相関係数が必要

h=my_lev_durb(order,r);//予測係数導出

h_update=[h',zeros(1,s_size-order-2)];

h_fft=fft(h_update);

spectrum_h=-20*log10(abs(h_fft));//包絡線

plot(x_scaling,spectrum(1:s_size/2));//信号のスペクトル描画

plot(x_scaling,spectrum_h(1:s_size/2),'r');//包絡線描画
g=gce();
c=g.children;
c.thickness=1.5;
xgrid();
title('LPC_Spectral_Envelope','fontsize',4);

```