［Scilab で信号処理 2010］

情報工学科　教授　幹　　康
ymiki@cs.takushoku-u.ac.jp

©2013 Yasushi Miki　　ver.13.07

[7] 音声分析（線形予測）

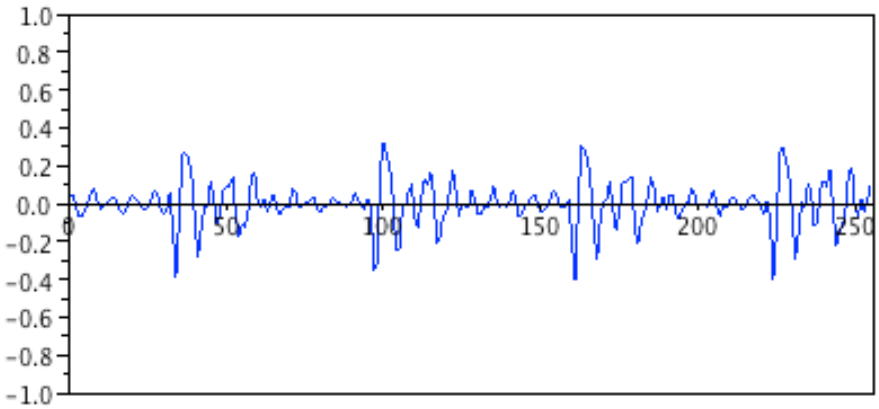
［以下に示す青色のコマンドは Scilab のコンソール上で実行できます．］

■ 音声(母音)の周波数分析

「あーーー」という音声(母音 /あ/)の周波数分析をしましょう.データの取得法はこれまでと同様です．
音声情報は基本的に 4kHz 以下に集約されていますから,以下では，8kHz でサンプリングした音声データを使用することにします．
前節と同様、プリエンファシス処理により、スペクトルの平坦化を図ります。

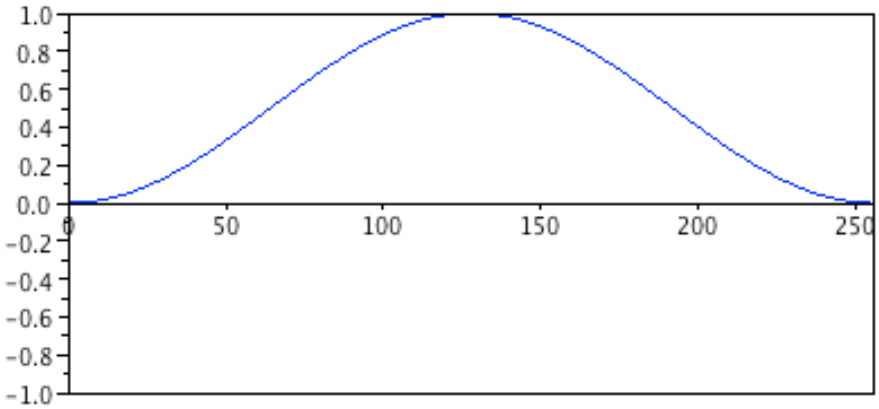
```
x = wavread('a_8k.wav');
//  プリエンファシス
y1 = x(1:256);          // x(n) に対応
y2 = x(2:257);          // x(n+1) に対応
y = y2 - 0.98*y1;       // 差分演算
//
plot(y);
n = 0:255;
clf();
plot2d(n,y,frameflag = 1,axesflag = 1, rect=[0,-1,256,1], style = 2);
//plot2d(n,y,frameflag = 2, axesflag = 1, style = 2);
f = gcf(); a = gca();
f.figure_position = [800,40];
f.figure_size = [480,320];
a.x_location = "origin";
```

波形を示します．



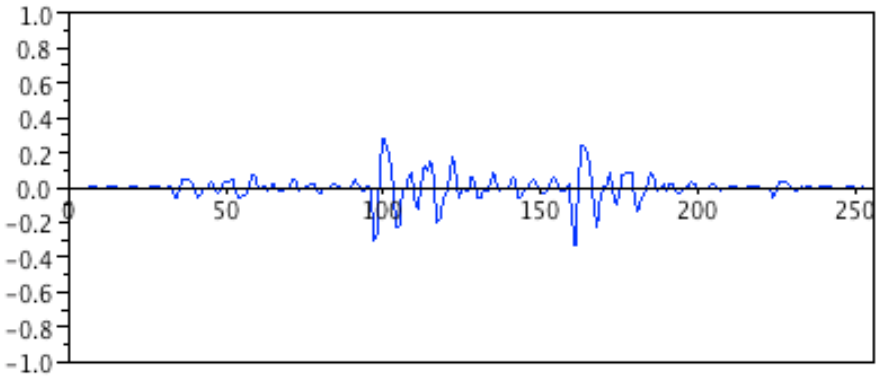
ハニング窓を作成します.

```
han = (1 - cos(2 * %pi * n / 256)) / 2;
scf();
plot2d(n, han, frameflag = 1,axesflag = 1, rect=[0,-1,256,1], style = 2);
```



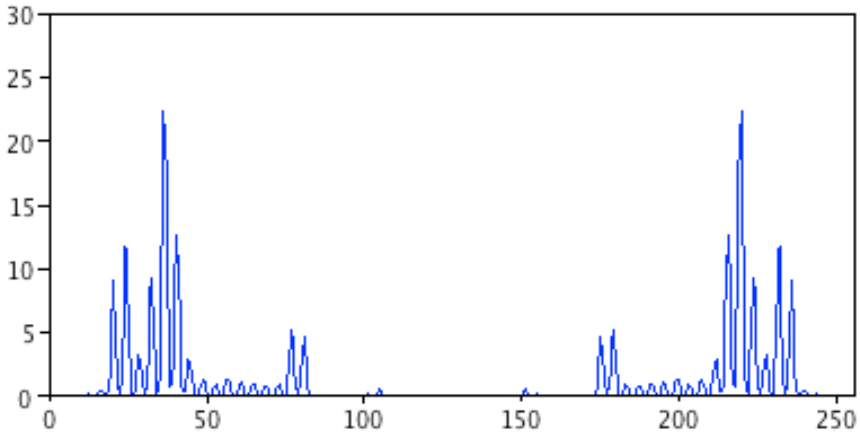
取得した波形にハニング窓を掛けます.

```
yh = y .* han;
//配列の要素ごとのかけ算は '.*' を使用します.
scf();
plot2d(n, yh, frameflag = 1,axesflag = 1, rect=[0,-1,256,1], style = 2);
```



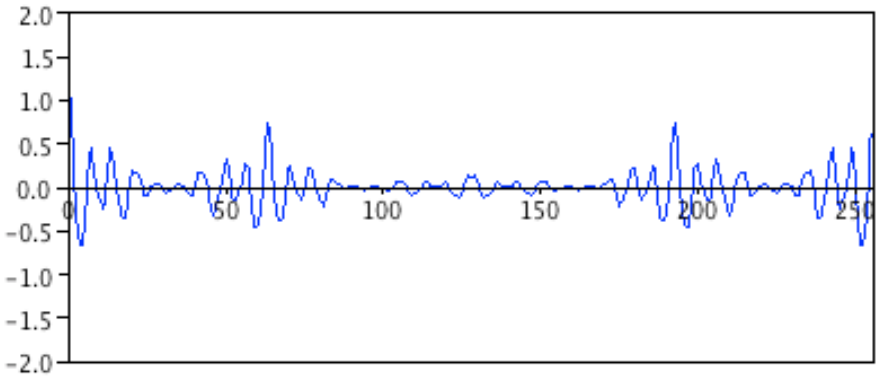
ハニング窓を掛けた信号をフーリエ変換し,ピリオドグラムを求めると以下のようになります.
（ピリオドグラムとは,スペクトルの絶対値の2乗で与えられます.広義のパワースペクトル）

```
scf();
spec = fft(yh);
periodogram = (abs(spec))^2;
plot2d(n,periodogram, rect=[0,0,256,50], frameflag = 1, axesflag = 1, style = 2)
```



ピリオドグラムをフーリエ逆変換することにより,自己相関関数φを計算します. ここでは,相関関数をN=256点計算していますが,実際に線形予測に利用される点数は20点程度です. (自己相関関数については教科書「ディジタル信号処理」を参照してください.)

```
scf();
fai=real(fft(periodogram/length(periodogram)));
plot2d(n,fai,rect=[0,-3,256,3], frameflag = 1, axesflag = 1, style = 2)
```



□線形予測係数の算出

以下のプログラムで線形予測係数を1次から p次まで,順次計算します.

1)Scilab の「アプリケーション」メニューから,「エディタ」を開き,

2)以下のプログラムを作成し,適当な名前で保存する.拡張子は.sce となる. たとえば. 'LP.sce'.

3)このファイルを,Scilab の「ファイル」メニューから「実行...」を選択し,実行する. あるいは,

```
exec('LP.sce');
```

としても良い.

```
p =20;
W = zeros(1,p+1);
E = zeros(1,p+1);
A=zeros(p+1,p+1);
```

```
W(1) = fai(2);
E(1) = fai(1);
for k = 1:p,
   A(k+1,k+1) = -W(k)/E(k);
   E(k+1) = E(k)*(1 - A(k+1,k+1)^2);
   for i = 1:k-1,
      A(k+1,i+1) = A(k,i+1) + A(k+1,k+1)*A(k,k-i+1);
    end
   W(k+1) = fai(k+2) ;
   for i = 1:k,
      W(k+1)=W(k+1)+ A(k+1,i+1)*fai(k+2-i);
   end
   printf('%f \t%f \t%f\n', A(k+1,k+1), E(k+1), W(k+1));
end
```

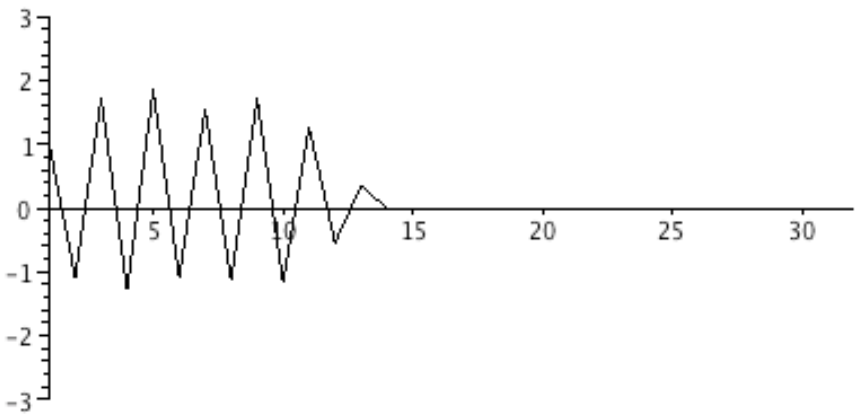
□モデルの次数の決定

上で p = 20 次までのモデルに対応する線形予測係数を求めましたが,どのモデルを採用するかを決定しなければなりません.その判定基準として「AIC基準」と呼ばれる計算法がありますが,ここでは,実用的観点から p = 12 に固定することにします.

線形予測係数は線形予測モデルのインパルス応答を与えるものと考えることができます.つまり,インパルス応答は A(p+1,i) (i=1:p+1) ただし,A(p+1,1)=1
となります.

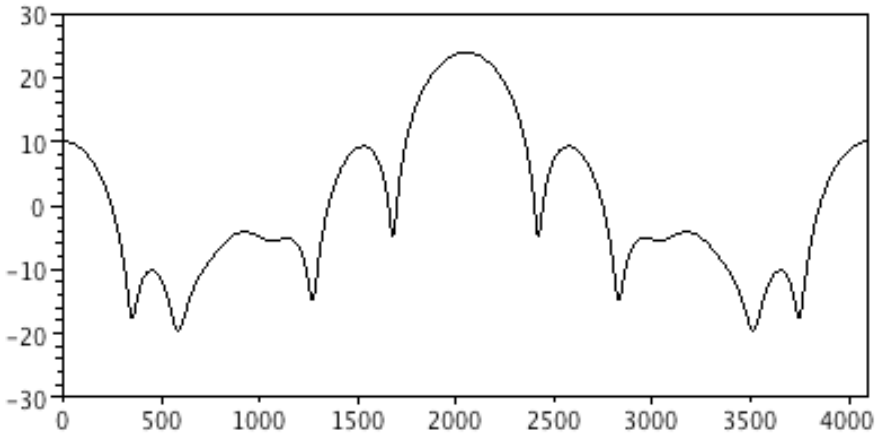
図示すると

```
p = 12;
 B = zeros(1, p+1);
 A(p+1,1)=1;
 for i = 1:p+1,
  B(i) = A(p+1,i);
 end
 scf();
 plot2d(1:32,resize_matrix(B,1,32), rect = [1,-4,32,4]);
```



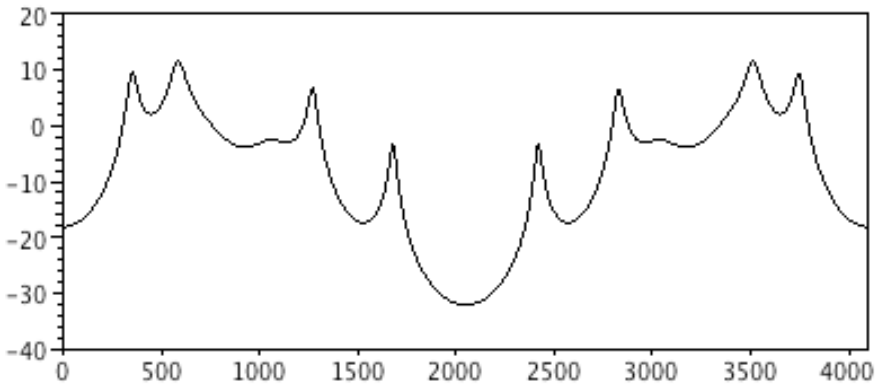
これをフーリエ変換すると,線形予測モデルの周波数特性となります.周波数刻みを細かくするために,インパルス応 答の長さを4096点に増やします.

```
B = resize_matrix(B,1,4096);
specB = fft(B);
scf();
plot2d(1:length(B),20*log10(abs(specB)),rect=[0,-30,4096,30])
```



この逆特性が自己回帰モデルの周波数特性になります。これが音声のスペクトル(包絡)になります。
配列の除算は ./ を使います。

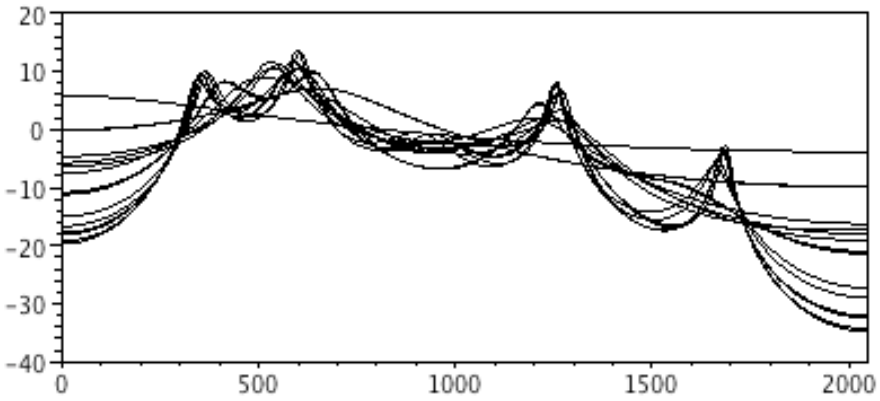
```
scf();
autoreg = sqrt(E(p+1))./specB;
plot2d(1:length(B),20*log10(abs(autoreg)),rect=[0,-40,4096,20])
```



ここで整理しておきます。
線形予測モデル: 音声信号を入力すると、線形予測残差信号が出力されるモデル。
自己回帰モデル: 予測残差信号を入力すると、音声信号が出力されるモデル。線形予測モデルの逆モデル。
モデルの次数を変化させて周波数特性の変化を見ると、以下のようになります。

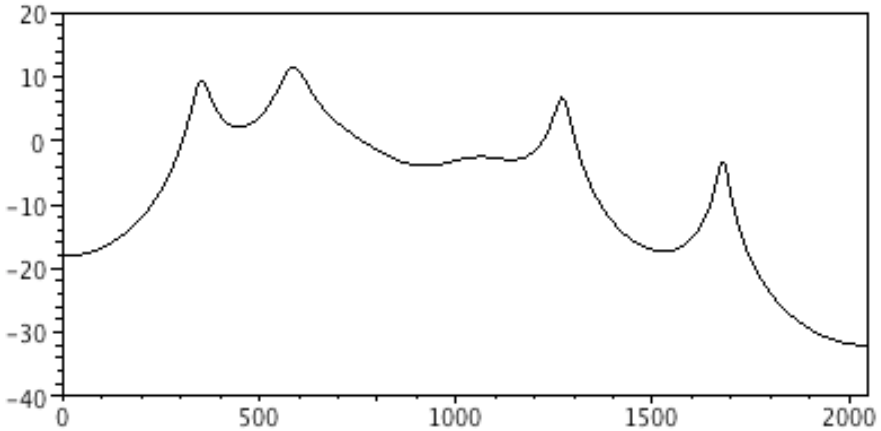
```
scf();
B = zeros(1,4096);
for p = 1:15,
```

```
  A(p+1,1) = 1;
  for i = 1:p+1,
    B(i) = A(p+1,i);
  end
  specB = fft(B);
  autoreg = sqrt(E(p+1))./specB;
  autoreg = resize_matrix(autoreg,1,2048);
  plot2d(1:2048,20*log10(abs(autoreg)),rect=[0,-40,2048,20])
end
```



次数 p = 12 のモデルの周波数特性を改めて示します。

```
scf();
  B = zeros(1,4096);
  p = 12;
A(p+1,1) = 1;
for i = 1:p+1,
    B(i) = A(p+1,i);
  end
specB = fft(B);
    autoreg = sqrt(E(p+1))./specB;
    autoreg = resize_matrix(autoreg,1,2048);
    plot2d(1:2048,20*log10(abs(autoreg)),rect=[0,-40,2048,20])
```



■フォルマント周波数の推定 上のグラフのピーク値から、フォルマント周波数を求めましょう。ピークは次のようにして求められます。3番目の ピークは小さいので無視することにします。

```
sp = 20*log10(abs(autoreg));
for i=2:length(sp)-1,
 if ((sp(i+1)-sp(i))<0 & (sp(i)-sp(i-1))>0) then printf("%d\t%f\n",i,
(i-1)*8000/4096);
end
end
```

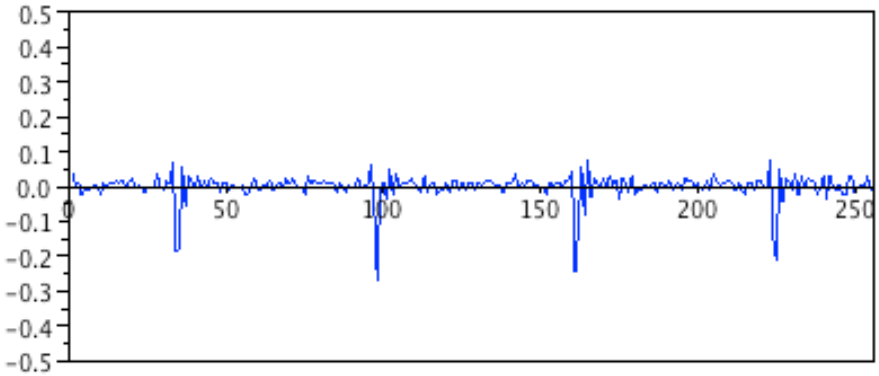
N	周波数
352	685.546875
586	1142.578125
1061	2070.312500
1270	2478.515625
1679	3277.343750

これから、第1、第2、第3フォルマント周波数は 686, 1143, 2479 Hz となります。

線形予測モデルは、音声信号を入力すると予測残差信号が出力として現れます。

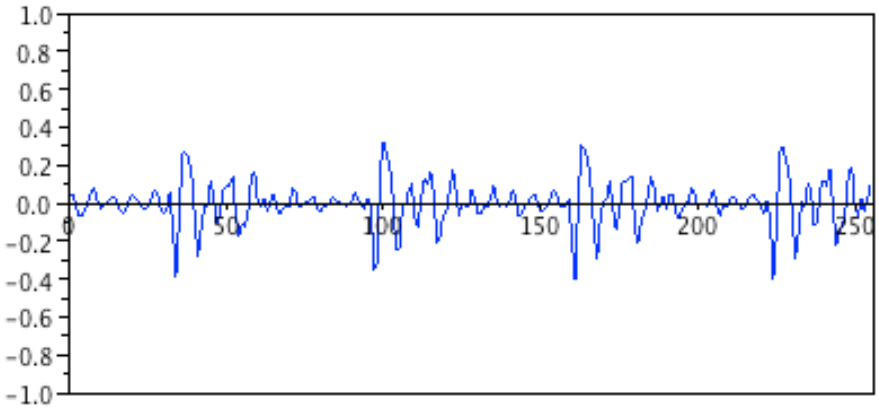
```
p=12;
yz = [zeros(1,p+1),y];
er = zeros(1,256);
for i = 1:256,
   for j = 1:p+1,
      er(i) = er(i) + yz(i + p+2 -j)*A(p+1,j);
end end
scf();
plot2d(1:256, er, frameflag = 1,axesflag = 1, rect=[0,-1,256,1], style = 2)
```

予測残差信号



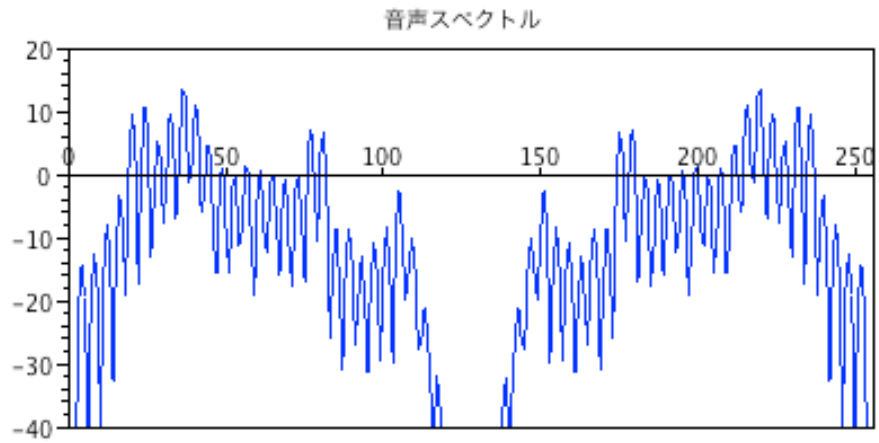
音声のピッチに対応するインパルス列が確認できます。元の音声波形と比べてみましょう。

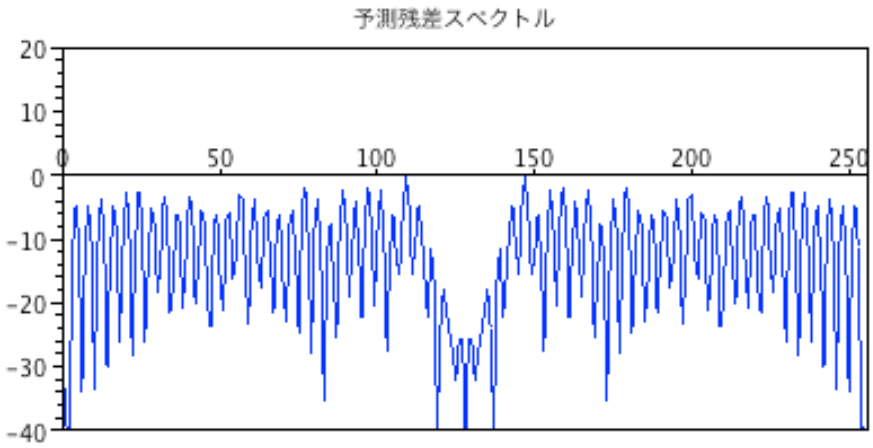
音声信号



スペクトルで比較してみましょう。

```
 spec = fft(yh);
 scf(); f=gcf();
f.figure_position = [800,40+f.figure_id*20];
f.figure_size = [480,320];
 plot2d(n,20*log10(abs(spec)), rect=[0,-40,256,20], frameflag = 1, axesflag = 1,
style = 2)
sper =fft(er .* han);
a=gca(); a.title.text = "音声スペクトル";
 scf();
 plot2d(n,20*log10(abs(sper)), rect=[0,-40,256,20], frameflag = 1, axesflag = 1,
style = 2)
a=gca(); a.title.text = "予測残差スペクトル";
```

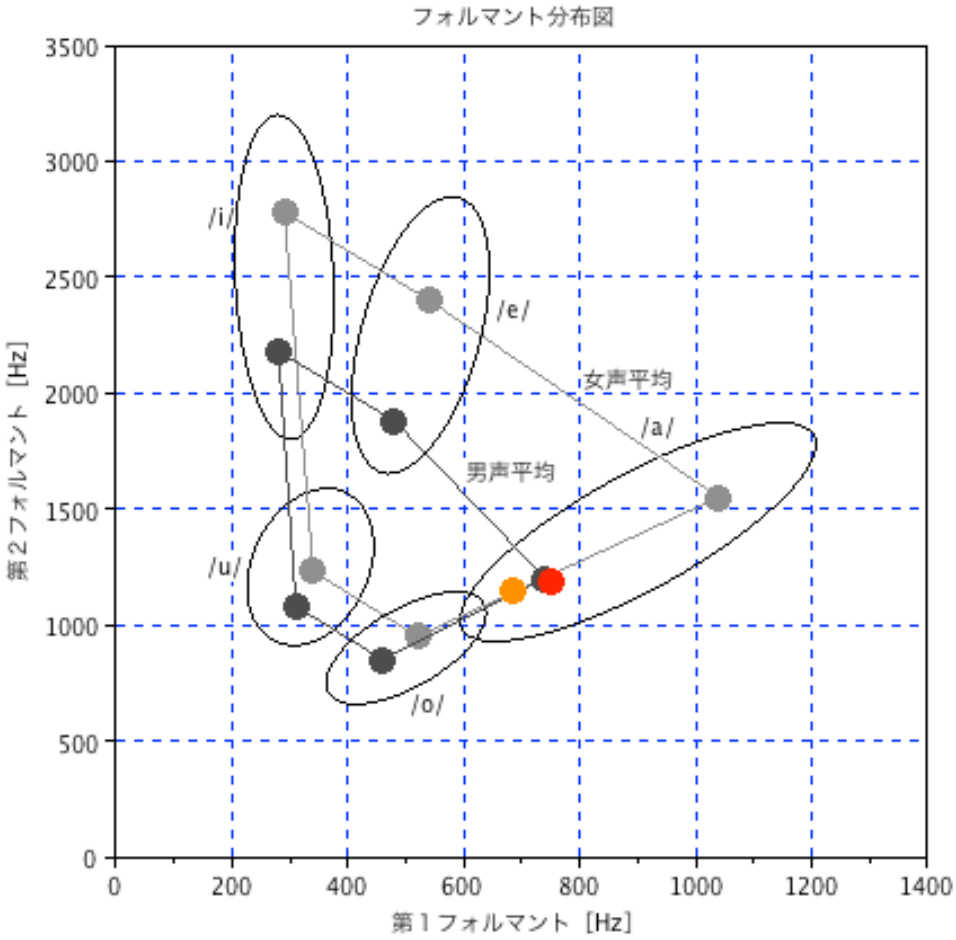




予測残差スペクトルは、ほぼ、平坦なスペクトルになっています。つまり、白色雑音です。白色であるということは、
まったく予測不能な成分であるということを意味します。しかしながら、音声のピッチに対応する周期的なスペクトル
の微細構造は保持されています。

最後に、この方法で求めたフォルマントと前回のケプストラム法で求めたものと比較してみましょう。
Scilab で信号処理(6)で作成したフォルマント分布図を再度作成し,その上に示します。

```
scf();
exec('FormantMap.sce');
F1=750;F2=1187; //ケプストラム法のフォルマント
plot2d(F1,F2)
a=gce();c=a.children;
c.mark_style=0;c.mark_size=1;c.mark_foreground=color(255,0,0);
F1=686;F2=1143; //線形予測法のフォルマント
plot2d(F1,F2)
a=gce();c=a.children;
c.mark_style=0;c.mark_size=1;c.mark_foreground=color(255,128,0);
```



赤い点がケプストラム法、オレンジの点が線形予測法による計算結果です。両者の違いについてどう考えるかは読者に任せます。