

【線形予測分析 (LPC) Linear Prediction】

ある信号の過去サンプル $x(n-k)$ $k = 1, 2, \dots$ から次の信号 $x(n)$ を予測する。
この時、予測値は次のように書ける。

$$\text{予測値: } \hat{x}(n) = - \sum_{k=1}^M h(k)x(n-k)$$

ここで、サンプル数は M 個であり、 $h(k)$ は線形予測係数である。
次に、予測値 $\hat{x}(n)$ と実際の値 $x(n)$ の誤差は、

$$e(n) = x(n) - \hat{x}(n) = x(n) + \sum_{k=1}^M h(k)x(n-k)$$

である。更に、誤差二乗和は、

$$\epsilon = \sum_{i=1}^M e^2(i) = \sum_i x^2(i) + 2 \sum_i \sum_k h(k)x(i)x(i-k) + \sum_i \sum_k \sum_m h(k)h(m)x(i-k)x(i-m)$$

上式 ϵ を最小にするために、予測係数 $h(j)$ ($j = 1, 2, \dots, M$) で偏微分し、その値を 0 とすると、

$$\frac{\partial \epsilon}{\partial h(j)} = 2 \sum_i x(i)x(i-j) + 2 \sum_i \sum_k h(k)x(i-k)x(i-j) = 0$$

ここから、次の等式が成立。

$$\sum_k h(k) \sum_i x(i-k)x(i-j) = - \sum_i x(i)x(i-j)$$

ここで、自己相関関数を用いて表現すると、以下のユール-ウォーカー方程式 (正規方程式) を得る。

$$\text{YuleWalker-Equations: } \sum_k h(k)r_{xx}(k-j) = -r_{xx}(j) \quad (j = 1, 2, \dots, M)$$

なお、自己相関関数は、 $r_{xx}(k) = \frac{1}{M} \sum_{n=1}^M x(n)x(n+k)$ である。

ユール-ウォーカー方程式を行列で表現すると、

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(M-1) \\ r_{xx}(1) & r_{xx}(0) & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ r_{xx}(M-1) & \cdots & r_{xx}(1) & r_{xx}(0) \end{bmatrix} \begin{bmatrix} h(1) \\ h(2) \\ \cdot \\ \cdot \\ h(M) \end{bmatrix} = - \begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \\ \cdot \\ \cdot \\ r_{xx}(M) \end{bmatrix}$$

左辺逆行列をかけることで予測係数を求めることができる。

また、左辺の行列はテプリッツ行列 (Toeplitz Matrix) であり、レビンソン・ダービン法 (Levinson-Durbin Algorithm) で再帰的に効率よく解ける。

■前向き線形予測 Forward Prediction

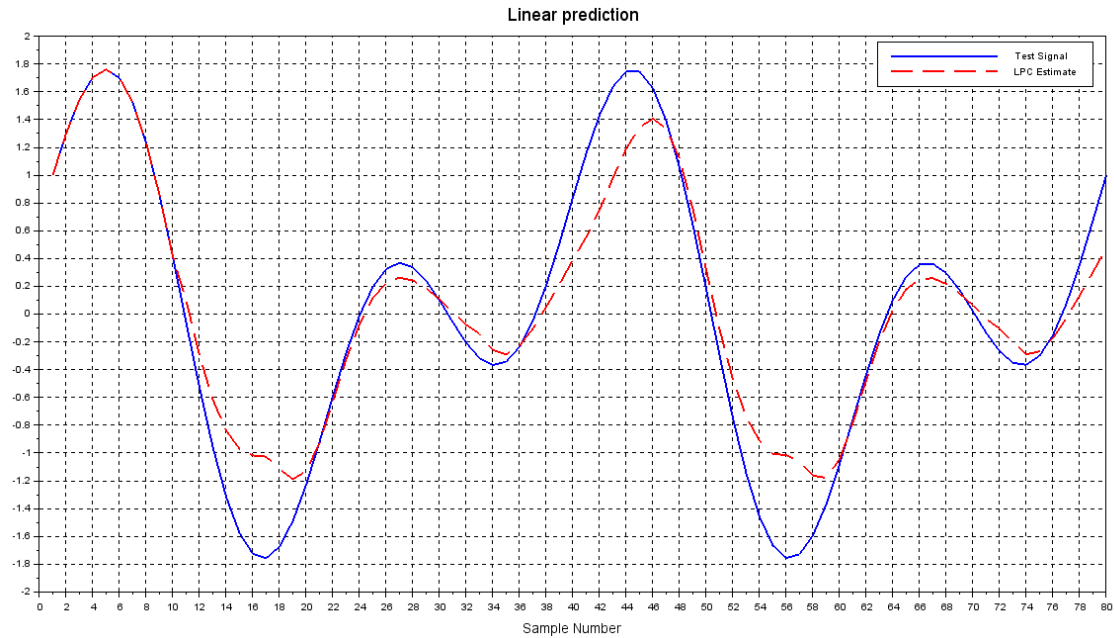


Figure 1: Scilab 実行結果
10次前向き線形予測

Source Code 1: Scilab

```
////////////////////////////////////  
//  
// 前向き線形予測  
// Forward Prediction  
//  
// M.Tsutsui  
////////////////////////////////////  
  
////////////////////////////////////  
//  
//  
// 前向き線形予測  
// 線形予測係数:正規方程式(Yule-Walker方程式)より,自己相関関数行列の逆行列を用いて求める  
//  
//  
////////////////////////////////////  
  
clear all;  
  
d_size=80; //data size  
x=linspace(0,4*%pi,d_size);  
d_signal=sin(2*x)+cos(x); //実測値(テスト信号)  
  
order=10; //次数  
  
est_signal=d_signal(1:order)'; //線形予測値  
  
funcprot(0); //自己相関関数  
function[res]=auto_corr(k,signal); //k:ラグ signal:対象信号(列ベクトルとする)  
  
[size_l,size_r]=size(signal);  
  
loop=0;
```

```

res=[];

for k=0:1:k-1;//ラグ 0~k-1まで
    for n=1:1:size_r-k;
        if(n+k>size_r)
            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);//ユーロウォーカー方程式 右辺 行ベクトル
function[res]=ywe_right_side(k,signal);//k: ラグ signal: 対象信号

[size_l,size_r]=size(signal);

loop=0;
res=[];

for k=1:1:k;
    for n=1:1:size_r-k;
        if(n+k>size_r)
            signal(n+k)=0;
        end
        loop=loop+signal(n)*signal(n+k);
    end

    res=[res;loop];

    loop=0;//値初期化

end

endfunction

for j=1:1:d.size-order;

    signal=d_signal(j:j+order-1);//サンプル移動

    h=inv(toeplitz(auto_corr(order,signal)))*ywe_right_side(order,signal);//予測係数 ライブラリ"toeplitz"を使用
    h_rev=flipdim(h,1);//予測係数 反転

    est_signal(order+j)=sum(h_rev.*signal ');//予測値

end

plot(d_signal);
a=gce();
c=a.children;
c.thickness = 2;

plot(est_signal,'r--');
a=gce();
c=a.children;
c.thickness = 2;

```

```
xgrid();
xlabel('Sample Number','fontsize',3);
legend(['Test Signal','LPC Estimate',]);
title('Linear prediction','fontsize',4);
```

■ 後向き線形予測 Backward Prediction

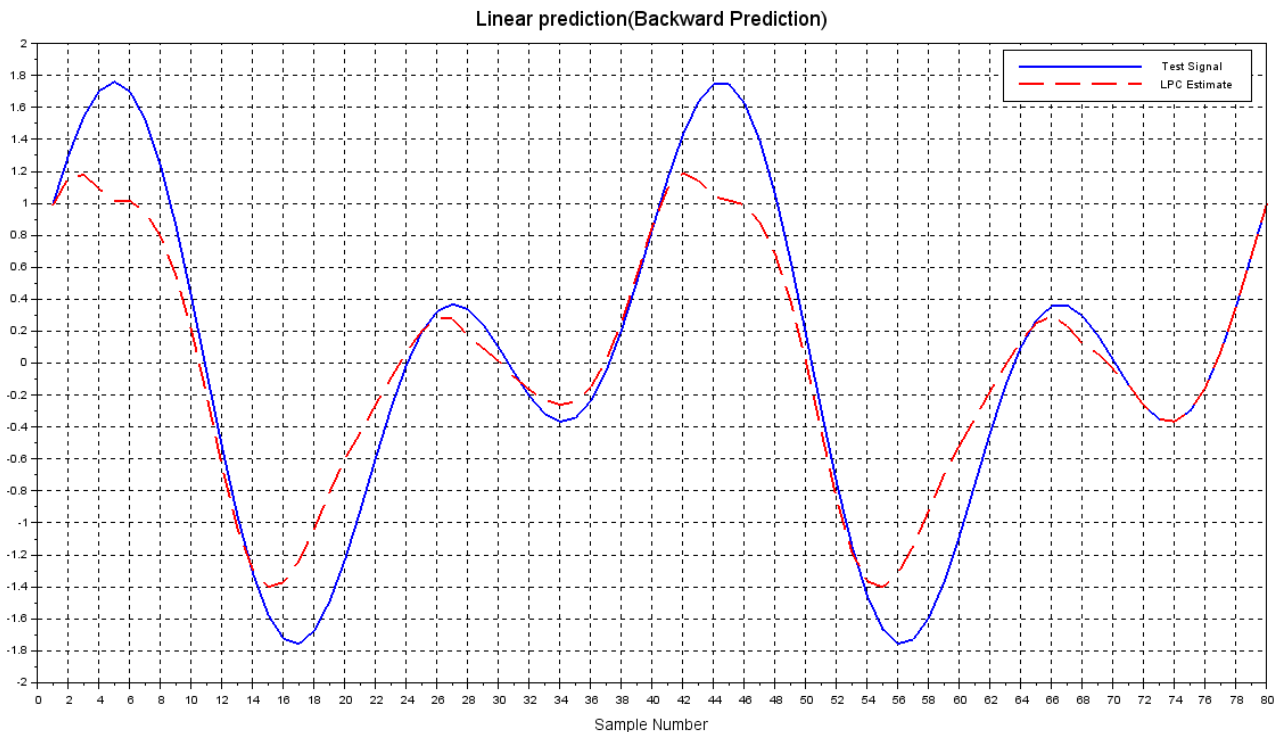


Figure 2: Scilab 実行結果
10 次後向き線形予測

Source Code 2: Scilab

```
////////////////////////////////////
// 後向き線形予測
// Backward Prediction
//
// M.Tsutsui
////////////////////////////////////
////////////////////////////////////
//
//
// 後向き線形予測
// 線形予測係数:正規方程式(Yule-Walker方程式)より,自己相関関数行列の逆行列を用いて求める
//
//
////////////////////////////////////

clear all;

d_size=80; //data size
x=linspace(0,4*%pi,d_size);
d_signal=sin(2*x)+cos(x); //実測値(テスト信号)

order=10; //次数
```

```

est_signal=flipdim(d_signal(d_size-order+1:d_size)',1);//線形予測値

funcprot(0);//自己相関関数
function[res]=auto_corr(k,signal);//k: ラグ signal: 対象信号(列ベクトルとする)

    [size_l,size_r]=size(signal);

    loop=0;
    res=[];

    for k=0:1:k-1;//ラグ 0~k-1まで
        for n=1:1:size_r-k;
            if(n+k>size_r)
                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);//ユールウォーカー方程式 右辺 行ベクトル
function[res]=ywe_right_side(k,signal);//lag: ラグ signal: 対象信号

    size=length(signal);

    loop=0;
    res=[];

    for k=1:1:k;
        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

for j=1:1:d_size-order;

    signal=flipdim(d_signal(d_size-order-j+2:d_size-j+1),2);//サンプル移動,順番反転

    h=inv(toeplitz(auto_corr(order,signal)))*flipdim(ywe_right_side(order,signal),1);//予測係数 ライブラリ"toeplitz"を使用

    est_signal(order+j)=sum(h.*signal ');//予測値
end

est_signal=flipdim(est_signal,1);//順番反転

plot(d_signal);
a=gce();
c=a.children;

```

```
c.thickness = 2;

plot(est_signal, 'r--');
a=gce();
c=a.children;
c.thickness = 2;
xgrid();
xlabel('Sample_Number', 'fontsize', 3);
legend(['Test_Signal'; 'LPC_Estimate'];);
title('Linear_prediction(Backward_Prediction)', 'fontsize', 4);
```