A FAST ALGORITHM FOR THE TWO-DIMENSIONAL COVARIANCE METHOD OF LINEAR PREDICTION

S. Lawrence Marple Jr.
Acuson Corp.
Mountain View, CA 94043

ABSTRACT

This paper presents a new fast computational algorithm for the solution of the least squares normal equations of the two-dimensional (2-D) covariance method of linear prediction. The fast algorithm exploits the near-to-doubly-Toeplitz structure of the normal equations when expressed in matrix form. This algorithm is useful for generating high resolution imagery from coherent imaging system in-phase/quadrature (I/Q) data, such as synthetic aperture radar (SAR).

1. CLASSICAL FFT IMAGE GENERATION

The classical imaging algorithm for 2-D coherent imaging system data $x[n_1, n_2]$, assumed available over the rectangular data grid $0 \leq n_1 < N_1$, $0 \leq n_2 < N_2$, is simply the squared magnitude of the 2-D DFT of the full data array $|X(f_1, f_2)|^2$ in which the 2-D DFT is

$$X(f_1, f_2) = \frac{1}{N_1 N_2} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} w[n_1, n_2] x[n_1, n_2] \exp(-j2\pi[f_1\, n_1\, \Delta T_1 + f_2\, n_2\, \Delta T_2]) \quad (1)$$

and $w[n_1, n_2]$ is an optional 2-D window function used to suppress sidelobes (at the expense of broadening the mainlobe response,thereby decreasing the spatial resolution) and ΔT_1 and ΔT_2 are the sampling intervals in the two dimensions.

However, the classical approach is limited in resolution. To achieve higher resolution in the 2-D image formed from the coherent 2-D phase data (SNR permitting), one approach is to use 2-D autoregressive (AR) spectral analysis, which may select least squares linear prediction techniques to estimate the AR parameters. The tradeoff one makes, of course, is a significant increase in the computations (relative to the FFT) in exchange for the potential resolution enhancement. Thus, fast algorithms are needed to reduce this additional computational burden.

2. 2-D AR PSD AND LINEAR PREDICTION

To develop the 2-D version of the 2-D AR spectrum (which forms the image) based on estimation of the 2-D AR parameters via the 2-D covariance method of linear prediction, consider the case of a 2-D first-quadrant (Q_1) quarter-plane linear prediction error filter which, for input 2-D signal $x[n_1, n_2]$ and $[p_1, p_2]$-th order linear prediction error filter with parameters $a^1[k_1, k_2]$ defined over $0 \leq k_1 \leq p_1$ and $0 \leq k_2 \leq p_2$, has the scalar linear prediction error

$$e^1[n_1, n_2] = \sum_{k_1=0}^{p_1} \sum_{k_2=0}^{p_2} a^1[k_1, k_2] x[n_1 - k_1, n_2 - k_2] \quad (2)$$

in which $a^1[0,0] = 1$ by definition. In anticipation of the fast computational algorithm to be presented, we shall assume that p_1, the row dimension, is the fixed order parameter and p_2, the column dimension, is the variable order parameter. An alternative block vector representation of the Q_1 quarter-plane linear prediction error filter output is

$$e^1[n_1, n_2] = \underline{\mathbf{a}}^1 \underline{\mathbf{x}}[n_1, n_2] \quad (3)$$

where

$$\underline{\mathbf{a}}^1 = \begin{pmatrix} \mathbf{a}^1[0] & \mathbf{a}^1[1] & \ldots & \mathbf{a}^1[p_2] \end{pmatrix}$$

is a block vector of block dimension $1 \times (p_2 + 1)$ with vector elements

$$\mathbf{a}^1[p] = \begin{pmatrix} a^1[0, p] & a^1[1, p] & \ldots & a^1[p_1, p] \end{pmatrix}$$

of dimension $1 \times (p_1 + 1)$ for $0 \leq p \leq p_2$, and

$$\underline{\mathbf{x}}^T[n_1, n_2] = [x[n_1, n_2] \ldots x[n_1 - p_1, n_2] \cdots x[n_1, n_2 - p_2] \ldots x[n_1 - p_1, n_2 - p_2]]$$

is a data vector of dimension $(p_1+1)(p_2+1) \times 1$. Recalling that the spectral density relationship in 2-D between input and output signal processes is

$$P_{\text{out}}(f_1, f_2) = |H(f_1, f_2)|^2 P_{\text{in}}(f_1, f_2), \quad (4)$$

in which $H(f_1, f_2)$ is the Fourier transform of the system function relating input and output, and assuming that $e^1[n_1, n_2]$ is a white noise process with variance $\rho^1 = \mathcal{E}\{|e^1[n_1, n_2]|^2\}$ so that $x[n_1, n_2]$ is a Q_1 quarter-plane two-dimensional autoregressive (2-D AR) process, then the Q_1 quarter-plane 2-D AR spectrum is given as

$$P^1(f_1, f_2) = \frac{\Delta T_1\, \Delta T_2\, \rho^1}{\left|\sum_{k_1}^{p_1} \sum_{k_2}^{p_2} a^1[k_1, k_2] \exp(-j2\pi[f_1\, k_1\, \Delta T_1 + f_2\, k_2\, \Delta T_2])\right|^2}$$

In a similar manner, one can define the Q_2, Q_3, and Q_4 quarter-plane linear prediction error filter outputs and their corresponding quarter-plane spectra $P^2(f_1, f_2)$ and $P^3(f_1, f_2)$ and $P^4(f_1, f_2)$.

3. 2-D NORMAL EQUATIONS

The 2-D least squares normal equations of the 2-D covariance method of linear prediction are obtained by assuming that 2-D data is available only over the intervals $0 \leq n_1 \leq N_1 - 1$ and $0 \leq n_2 \leq N_2 - 1$, so that valid linear prediction errors $e^i[n_1, n_2]$ for quadrants $i = 1, 2, 3, 4$ can only be formed over the intervals $p_1 \leq n_1 \leq N_1 - 1$ and $p_2 \leq n_2 \leq N_2 - 1$ without running off the ends of the data. The total squared error then becomes

$$\begin{aligned} \rho^i &= \sum_{n_1=p_1}^{N_1-1} \sum_{n_2=p_2}^{N_2-1} \left| e^i[n_1, n_2] \right|^2 \\ &= \underline{\mathbf{a}}^i \Big(\sum_{n_1=p_1}^{N_1-1} \sum_{n_2=p_2}^{N_2-1} \underline{\mathbf{x}}[n_1, n_2] \underline{\mathbf{x}}^H[n_1, n_2] \Big) \underline{\mathbf{a}}^{iH} \\ &= \underline{\mathbf{a}}^i \underline{\mathbf{R}}\, \underline{\mathbf{a}}^{iH} \end{aligned} \tag{5}$$

in which the matrix $\underline{\mathbf{R}}$ of dimension $(p_1 + 1)(p_2 + 1) \times (p_1 + 1)(p_2 + 1)$ has the alternative representations

$$\begin{aligned} \underline{\mathbf{R}} &= \sum_{n_1=p_1}^{N_1-1} \sum_{n_2=p_2}^{N_2-1} \underline{\mathbf{x}}[n_1, n_2] \underline{\mathbf{x}}^H[n_1, n_2] \\ &= \underline{\mathbf{X}}\, \underline{\mathbf{X}}^H \\ &= \begin{pmatrix} \mathbf{R}[0,0] & \mathbf{R}[0,1] & \dots & \mathbf{R}[0,p_2] \\ \mathbf{R}[1,0] & \mathbf{R}[1,1] & \dots & \mathbf{R}[1,p_2] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}[p_2,0] & \mathbf{R}[p_2,1] & \dots & \mathbf{R}[p_2,p_2] \end{pmatrix} \end{aligned}$$

where

$$\mathbf{R}[i,j] = \sum_{n_2=p_2}^{N_2-1} \mathbf{X}[n_2 - i] \mathbf{X}^H[n_2 - j]$$

are matrix elements of dimension $(p_1 + 1) \times (p_1 + 1)$,

$$\underline{\mathbf{X}} = \begin{pmatrix} \mathbf{X}[p_2] & \mathbf{X}[p_2+1] & \dots & \mathbf{X}[N_2-1] \\ \mathbf{X}[p_2-1] & \mathbf{X}[p_2] & \dots & \mathbf{X}[N_2-2] \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{X}[0] & \mathbf{X}[1] & \dots & \mathbf{X}[N_2-p_2-1] \end{pmatrix}$$

is a rectangular block-Toeplitz 2-D data matrix of block dimension $(p_2 + 1) \times (N_2 - p_2)$, and $\mathbf{X}[k]$ is defined as

$$\begin{pmatrix} x[p_1,k] & x[p_1+1,k] & \dots & x[N_1-1,k] \\ x[p_1-1,k] & x[p_1,k] & \dots & x[N_1-2,k] \\ \vdots & \vdots & \ddots & \vdots \\ x[0,k] & x[1,k] & \dots & x[N_1-p_1-1,k] \end{pmatrix}$$

which is a rectangular Toeplitz 2-D data matrix of dimension $(p_1+1) \times (N_1 - p_1)$. Note that the total squared error of eq. (5) can be an estimate of the variance if normalized as $\rho^i / (N_1 - p_1)(N_2 - p_2)$.

If the total squared error is minimized, it can be shown that the resulting least squares normal equations take the form

$$\begin{aligned} \underline{\mathbf{a}}^1 \underline{\mathbf{R}} &= \begin{pmatrix} \rho^1 & \dots & 0 & 0 & \dots & 0 & \cdots & 0 & \dots & 0 \end{pmatrix} \\ \underline{\mathbf{a}}^2 \underline{\mathbf{R}} &= \begin{pmatrix} 0 & \dots & 0 & 0 & \dots & 0 & \cdots & \rho^2 & \dots & 0 \end{pmatrix} \\ \underline{\mathbf{a}}^3 \underline{\mathbf{R}} &= \begin{pmatrix} 0 & \dots & 0 & 0 & \dots & 0 & \cdots & 0 & \dots & \rho^3 \end{pmatrix} \\ \underline{\mathbf{a}}^4 \underline{\mathbf{R}} &= \begin{pmatrix} 0 & \dots & \rho^4 & 0 & \dots & 0 & \cdots & 0 & \dots & 0 \end{pmatrix} \end{aligned}$$

Unlike the Toeplitz-block-Toeplitz matrix of the known 2-D autocorrelation case (that is, the 2-D Yule-Walker equations), the least squares matrix does not share this property, although it is formed as the product of the rectangular Toeplitz-block-Toeplitz data matrix $\underline{\mathbf{X}}$. It does have hermitian symmetry, $\underline{\mathbf{R}} = \underline{\mathbf{R}}^H$. One can compute the four quadrant AR spectra and then form a single unbiased 2-D AR spectrum from the four individual 2-D AR spectra as follows

$$\frac{1}{P^{\text{combined}}(f_1, f_2)} = \frac{1}{P^1(f_1, f_2)} + \frac{1}{P^2(f_1, f_2)} + \frac{1}{P^3(f_1, f_2)} + \frac{1}{P^4(f_1, f_2)}$$

4. SOLUTION OF 2-D EQUATIONS

A fast computational algorithm for solution of $\underline{\mathbf{a}}^1$ to $\underline{\mathbf{a}}^4$ is not based on direct solution for the four quadrants of 2-D linear prediction/AR parameters, but is based on solving a special variant of the multichannel covariance algorithm involving the solution of the following set of multichannel least squares normal equations of order p and "time" index N_2

$$\begin{aligned} \underline{\mathbf{a}}_p \underline{\mathbf{R}}_p &= \begin{pmatrix} \mathbf{P}_p^a & 0 & \dots & 0 \end{pmatrix} \\ \underline{\mathbf{b}}_p \underline{\mathbf{R}}_p &= \begin{pmatrix} 0 & \dots & 0 & \mathbf{P}_p^b \end{pmatrix} \end{aligned}$$

where

$$\underline{\mathbf{R}}_p = \sum_{k=p}^{N_2-1} \underline{\mathbf{x}}[k] \underline{\mathbf{x}}^H[k]$$

and the block vectors of block dimension $1 \times (p + 1)$ are defined as

$$\underline{\mathbf{a}}_p = \begin{pmatrix} \mathbf{I} & \mathbf{A}_p[1] & \dots & \mathbf{A}_p[p] \end{pmatrix}$$

$$\underline{\mathbf{b}}_p = \begin{pmatrix} \mathbf{B}_p[p] & \dots & \mathbf{B}_p[1] & \mathbf{I} \end{pmatrix}$$

$$\underline{\mathbf{x}}_p[n] = \begin{pmatrix} \mathbf{X}[n] \\ \mathbf{X}[n-1] \\ \vdots \\ \mathbf{X}[n-p] \end{pmatrix}$$

Note that at $p = p_1$, $\underline{\mathbf{R}}_{p_1}$ is identical to $\underline{\mathbf{R}}$ in eq. (5), so one derives $\underline{\mathbf{a}}^i$ for $i = 1, 2, 3, 4$ from $\underline{\mathbf{a}}_{p_1}$ or $\underline{\mathbf{b}}_{p_1}$. For example

$$\mathbf{a}^1[0] = \begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix} \left[\mathbf{P}^a_{p_1}\right]^{-1}$$

and scaled such that $a^1[0,0] = 1$, as follows

$$\mathbf{a}^1[k] = \mathbf{a}^1[0]\mathbf{A}_{p_1}[k] \text{ for } 1 \leq k \leq p_1$$

and similarly

$$\mathbf{a}^4[0] = \begin{pmatrix} 0 & \dots & 0 & 1 \end{pmatrix} \left[\mathbf{P}^a_{p_1}\right]^{-1}$$

also scaled such that $a^4[0,0] = 1$

$$\mathbf{a}^4[k] = \mathbf{a}^4[0]\mathbf{A}_{p_1}[k] \text{ for } 1 \leq k \leq p_1.$$

Details of the fast algorithms for the multichannel covariance method of linear prediction and the two-dimensional covariance method of linear prediction may be found in the text *Digital Time, Frequency, and Spatial Analysis* by Marple (Prentice Hall, 1995).

5. MATLAB LISTING

```
function [p_Q1,a_Q1,p_Q2,a_Q2,p_Q3,a_Q3,p_Q4,a_Q4]...
                                = covar_2D(p1,p2,x)

% Two-dimensional quarter-plane support version of
% the covariance least squares linear prediction
% algorithm using QR-decomposition fast solution.
% p1 is the fixed order & p2 is the variable order.
% It is assumed that p2 >= p1 for most efficient
% computations, else switch roles of p1 and p2. All
% four quadrants (Q1,Q2,Q3,Q4) are computed simul-
% taneously by this fast computational algorithm.
%
% [p_Q1,a_Q1,p_Q2,a_Q2,p_Q3,a_Q3,p_Q4,a_Q4] ...
%                               = covar_2D(p1,p2,x)
%
% p1   -- row order of 2-D linear prediction/
%         AR filter
% p2   -- column order of 2-D linear prediction/
%         AR filter
% x    -- matrix of N1 x N2 2-D data samples:
%         x(row sample #,column sample #)
% p_Q1 -- least sqs. estimate of Q1 quarter-
%         plane linear prediction variance
% a_Q1 -- matrix of Q1 quarter-plane linear
%         prediction/AR 2-D parameters
% p_Q2 -- least sqs. estimate of Q2 quarter-
%         plane linear prediction variance
% a_Q2 -- matrix of Q2 quarter-plane linear
%         prediction/AR 2-D parameters
% p_Q3 -- least sqs. estimate of Q3 quarter-
%         plane linear prediction variance
% a_Q3 -- matrix of Q3 quarter-plane linear
%         prediction/AR 2-D parameters
% p_Q4 -- least sqs. estimate of Q4 quarter-
%         plane linear prediction variance
% a_Q4 -- matrix of Q4 quarter-plane linear
%         prediction/AR 2-D parameters

%****************  Initialization  ***************

[N1,N2] = size(x);
p = p1 + 1;
Np = N1 - p1;
if  p*(p2+1) > Np*(N2-p2)
    error('Orders p1 & p2 give solution singular.')
end
X = [];
for k=1:N2
    X = [X toeplitz(x(p:-1:1,k),x(p:N1,k))];
end
P = hermitian(X*X');
X1 = toeplitz(x(p:-1:1,1),x(p:N1,1));
XN = toeplitz(x(p:-1:1,N2),x(p:N1,N2));
p_a = hermitian(P - X1*X1');
p_b = hermitian(P - XN*XN');
a = [];
b = [];
c = XN'/P;         % use Toeplitz inversion ?
d = X1'/P;         % use Toeplitz inversion ?
ea = X;
eb = X;
ec = c*X;
ed = d*X;
I = eye(p,p);
II = eye(Np,Np);
Z = zeros(p,p);
ZZ = zeros(Np,p);

clear P X X1 XN

%****************  Main Recursion  ***************

for k=1:p2

disp(['Now at recursive iteration ',int2str(k)])
fix(clock)
    n = Np*(N2-k);

    % error condition checks
    if any(diag(p_a) <= 0) | any(diag(p_b) <= 0)
       error('Covariance matrix diag element <= 0')
    end
    gam = diag(hermitian(ec(:,n+1:n+Np)));
    del = diag(hermitian(ed(:,1:Np)));
    if any(gam < 0) | any(gam >= 1) | ...
                   any(del < 0) | any(del >= 1)
       error('Diag element gain factor not 0 to 1')
    end
```

```
% compute partial correlation and reflection
% coefficient matrices
ea = ea(:,Np+1:size(ea,2));
eb = eb(:,1:size(eb,2)-Np);
delta = ea*eb';
k_a = -delta/p_b;
k_b = -delta'/p_a;

% order updates for error covariance
% matrices  p_a and p_b
p_a = hermitian((I - k_a*k_b)*p_a);
p_b = hermitian((I - k_b*k_a)*p_b);

% order updates for linear prediction parameter
% arrays  a and b
temp = a;
a = [temp Z] + k_a*[b  I];
b = [Z b] + k_b*[I temp];

% check if maximum order has been reached
if k == p2, break, end

% order updates for prediction error
% arrays  ea and eb
temp = ea;
ea = temp + k_a*eb;
eb = eb + k_b*temp;

% square matrix coefficients for next updates
c1 =ec(:,1:Np);
c2 = c1/hermitian(II - ed(:,1:Np));
c3 = c1'/hermitian(II - ec(:,n+1:n+Np));

% time updates for gain vectors  c' and d"
temp = c;
c = temp + c2*d;
d = d + c3*temp;

% time updates of gain "errors"  ec' and ed"
temp = ec;
ec = temp + c2*ed;
ed = ed + c3*temp;
ec = ec(:,Np+1:size(ec,2));
ed = ed(:,1:size(ed,2)-Np);

% error condition checks
if any(diag(p_a) <= 0) | any(diag(p_b) <= 0)
   error('DIag element of a covar matrix <= 0')
end
gam = diag(hermitian(ec(:,n-Np+1:n)));
del = diag(hermitian(ed(:,1:Np)));
if any(gam < 0) | any(gam >= 1) | ...
                  any(del < 0) | any(del >= 1)
   error('Diag element gain factor not 0 to 1')
end

% rectangular matrix coefficients for next
% set of updates
c1 = ea(:,1:Np);
c2 = eb(:,n-Np+1:n);
c3 = c2'/p_b;
c4 = c1'/p_a;
c5 = c1/hermitian(II - ed(:,1:Np));
   c6 = c2/hermitian(II - ec(:,n-Np+1:n));

   % order updates for c and d; time
   % updates for a' and b"
   temp = a;
   a = temp + c5*d;
   d = [ZZ d] + c4*[I temp];
   temp = b;
   b = temp + c6*c;
   c = [c ZZ] + c3*[temp I];

   % time updates for p_a' and p_b"
   p_a = hermitian(p_a - c5*c1');
   p_b = hermitian(p_b - c6*c2');

   % order updates for ec and ed; time
   % updates for ea' and eb"
   temp = ed;
   ed = temp + c4*ea;
   ea = ea + c5*temp;
   temp = ec;
   ec = temp + c3*eb;
   eb = eb + c6*temp;
end

disp('Starting 2-D AR generation')
clear ea eb ec ed temp

%******* compute Q1 2-D AR parameter matrix *******

p_inv = [1 zeros(1,p1)]/p_a;
p_Q1 = 1/real(p_inv(1));
a_Q1 = p_Q1*p_inv;
a_Q1 = [a_Q1 a_Q1*a];
a_Q1 = reshape(a_Q1,p,p2+1);

%******* compute Q2 2-D AR parameter matrix *******

p_inv = [1 zeros(1,p1)]/p_b;
p_Q2 = 1/real(p_inv(1));
a_Q2 = p_Q2*p_inv;
a_Q2 = [a_Q2*b a_Q2];
a_Q2 = fliplr(reshape(a_Q2,p,p2+1));

%******* compute Q3 2-D AR parameter matrix *******

p_inv = [zeros(1,p1) 1]/p_b;
p_Q3 = 1/real(p_inv(p));
a_Q3 = p_Q3*p_inv;
a_Q3 = [a_Q3*b a_Q3];
a_Q3 =  flipud(fliplr(reshape(a_Q3,p,p2+1)));

%******* compute Q4 2-D AR parameter matrix *******

p_inv = [zeros(1,p1) 1]/p_a;
p_Q4 = 1/real(p_inv(p));
a_Q4 = p_Q4*p_inv;
a_Q4 = [a_Q4 a_Q4*a];
a_Q4 = flipud(reshape(a_Q4,p,p2+1));

% Copyright 1995
```