algorithm for both Gaussian and non-Gaussian inputs, the improvement over the LMS algorithm is greatest for large step sizes and short filter lengths.

REFERENCES


Wen-Hsien Fang and Andrew E. Yagle

Abstract—New fast algorithms for linear least squares smoothing problems in one and two dimensions are derived. These are discrete and multidimensional generalizations of the Bellman–Siegert–Krein resolvent identity, which has been applied to the continuous, one-dimensional stationary smoothing problem by Kailath. The new equations relate the linear least squares prediction filters associated with discrete random fields to the smoothing filters for those fields. This results in new fast algorithms for deriving the latter from the former. In particular, used in conjunction with recently developed generalized one- (two-) dimensional split Levinson and Schur algorithms for covariances with (block) Toeplitz-plus-Hankel structure, these algorithms can be used to compute smoothing filters for random fields defined on a polar raster, using fewer computations than those required by previous algorithms.

I. INTRODUCTION

In tomographic imaging problems solved by filtered back-projection [1], and in spotlight synthetic aperture radar [2], data are acquired on a polar raster of points, rather than on a rectangular lattice. Although it is possible to interpolate from the polar raster to a rectangular lattice, it is clearly preferable to deal with the data as it is. This is particularly true if the data are noisy, and smoothing is required.

Regarding the data as a random field with a known covariance function, linear least squares smoothing may be performed. Computation of the smoothing filter requires solution of two-dimensional discrete normal equations in polar coordinates. Fast algorithms for solving these equations are desirable when the covariance has some structure. However, properties such as stationarity are not manifested as block-Toeplitz structure when the random field is defined on a polar raster. For example, the covariance of an isotropic random field on a rectangular lattice is a Toeplitz function of the abscissae and ordinates, while on a polar raster it is a Toeplitz-plus-Hankel function of the radii.

Kailath [3] has noted the applicability of the Bellman–Siegert–Krein (BSK) resolvent identity to smoothing problems for continuous one-dimensional stationary random processes. First, the prediction filter for the process is computed, using the continuous-time Krein-Levinson equations, or by direct solution of the Wiener-Hopf integral equation. Then the BSK identity is used to compute the smoothing filter, which is the Fredholm resolvent of the integral operator associated with the covariance function. This approach has been extended to continuous-time close-to-Toeplitz covariances [4] and continuous-parameter isotropic random fields [5], although the latter uses a Fourier expansion into one-dimensional processes.

In this correspondence we generalize Kailath’s approach in three ways: 1) from continuous time to discrete time, resulting in an algorithm directly applicable to real discrete data; 2) from one dimension to two dimensions, without requiring an assumption of isotropy or an initial Fourier expansion; and 3) from stationary to nonstationary random fields.

Although the new algorithms of this correspondence do not require the covariance function to have special structure, they are most useful when used in conjunction with fast algorithms for computing the prediction filters that do require and exploit special structure in the covariance function. These include the Levinson algorithm [6] for stationary one-dimensional random processes, the algorithm of [7] for nonstationary one-dimensional random processes with Toeplitz-plus-Hankel covariances, and the algorithm of [8] for two-dimensional random fields on a polar raster with Toeplitz-plus-Hankel structure in the radial and angular variables of the covariance.

The correspondence is organized as follows. Section II derives
II. DERIVATION OF THE 1-D SMOOTHING FILTER

A. The Basic Problem

The smoothing problem considered in this section is as follows. Given noisy observations \( \{ y_k, -M \leq k \leq M \} \) of a zero-mean real-valued random process \( \{ x_k \} \), compute the linear least squares estimates of \( x_k \) for each \( k \) using all of the observations. The observations are related to the process by \( y_k = x_k + n_k \), where \( \{ n_k \} \) is zero-mean discrete white noise with unit power uncorrelated with \( \{ x_k \} \) (white noise with arbitrary power \( \sigma^2 \) can easily be handled by scaling). The covariance function \( k_{ij} = E[x_i x_j] \) of \( \{ x_k \} \) is known, and is assumed to be positive semidefinite.

The linear least squares estimate \( \hat{x}_k \) of \( x_k \), based on \( \{ y_k, -M \leq k \leq M \} \) can be expressed as

\[
\hat{x}_k = \sum_{j=-M}^{M} g_{kj} y_j
\]

where the superscript \( M \) for \( g_{kj} \) denotes that the range of the data is from \(-M\) to \( M\). Using the orthogonality principle of linear least squares estimation, the smoothing filters \( g_{kj} \) can be computed by solving the discrete normal equations

\[
k_{ij} = g_{ij}^M + \sum_{n=-M}^{M} g_{nj}^M k_{nj}
\]

for \(-M \leq i, j \leq M\) \hspace{1cm} (2)

In the special case when \( i = M + 1 \), (2) becomes the discrete Wiener–Hopf equation

\[
k_{M+1,j} = g_{M+1,j}^M + \sum_{n=-M}^{M} g_{n+1,j}^M k_{n+1,j}
\]

\[
= h_{M+1,j} + \sum_{n=-M}^{M} h_{n+1,j} k_{n+1,j}
\]

for \(-M \leq j \leq M\) \hspace{1cm} (3)

where \( h_{M+1,j} \equiv g_{M+1,j}^M \) is the prediction filter. The \( h_{ij} \) are assumed to have been already computed, presumably using some fast algorithm such as those of [6], [7], or [8]. Our objective is to derive a recursive formula for computing the smoothing filters \( g_{ij}^M \) from the previously computed prediction filters \( h_{ij} \).

B. Derivation of the Algorithm

Writing (2) with \( M \) replaced by \( M + 1 \) and subtracting (2) gives

\[
0 = (g_{ij}^{M+1} - g_{ij}^M) + \sum_{n=-M}^{M} (g_{n+1,j}^M - g_{n+1,j}^{M+1}) k_{nj}
\]

\[
+ g_{M+1,j}^M (k_{M+1,j} + h_{M+1,j}) + g_{M-1,j}^M (k_{M-1,j} + h_{M-1,j})
\]

\[
(4)
\]

Inserting (3) in (4) results in

\[
0 = (g_{ij}^{M+1} - g_{ij}^M) + \sum_{n=-M}^{M} (g_{n+1,j}^M - g_{n+1,j}^{M+1}) k_{nj}
\]

\[
+ g_{M+1,j}^M (k_{M+1,j} + h_{M+1,j}) + g_{M-1,j}^M (k_{M-1,j} + h_{M-1,j})
\]

and reordering (5) gives

\[
g_{ij}^{M+1} = g_{ij}^M - \sum_{n=-M}^{M} (g_{n+1,j}^M - g_{n+1,j}^{M+1}) k_{nj}
\]

\[
- \sum_{n=-M}^{M} \frac{g_{M+1,j}^M}{g_{n+1,j}^M} h_{M+1,j} - \sum_{n=-M}^{M} \frac{g_{M-1,j}^M}{g_{M+1,j}^M} h_{M-1,j}
\]

\[
(6)
\]

Since the covariance function \( k_{ij} \) is positive semidefinite by assumption, \( k_{ij} \) and \( h_{ij} \) are positive definite, and the solution to any system of equations with system matrix consisting of \( k_{ij} \) and \( h_{ij} \) must be unique. Therefore, we have

\[
g_{ij}^{M+1} = g_{ij}^M - \sum_{n=-M}^{M} (g_{n+1,j}^M - g_{n+1,j}^{M+1}) k_{nj}
\]

\[
- \sum_{n=-M}^{M} (g_{M+1,j}^M - g_{M+1,j}^{M+1}) h_{M+1,j} - \sum_{n=-M}^{M} (g_{M-1,j}^M - g_{M-1,j}^{M+1}) h_{M-1,j}
\]

\[
(7)
\]

Equation (7) allows \( g_{ij}^{M+1} \) to be computed recursively from \( g_{ij}^M \) and the prediction filters \( h_{M+1,j} \) and \( h_{M-1,j} \). Note that \( g_{ij}^M \) and \( h_{M+1,j} \) may be computed in parallel.

C. Computation of Boundary Points

In order to use (7), the boundary points \( g_{ij}^{M+1} \) must be computed first. This can be done as follows. Setting \( j = \pm (M + 1) \) in (2), we have

\[
\begin{align*}
\left[ 1 + k_{M+1,j}(M+1) \right] g_{ij}^{M+1} & = \sum_{n=-M}^{M} h_{n+1,j}(M+1) k_{nj}(M+1) \\
+ \left[ k_{-M+1,j}(M+1) \right] g_{ij}^{M+1} & = \sum_{n=-M}^{M} h_{n+1,j}(M+1) k_{nj}(M+1)
\end{align*}
\]

\[
(8)
\]

\[
\begin{align*}
\left[ k_{M+1,j}(-M-1) \right] g_{ij}^{M+1} & = \sum_{n=-M}^{M} h_{n+1,j}(-M-1) k_{nj}(-M-1) \\
+ \left[ 1 + k_{-M+1,j}(-M-1) \right] g_{ij}^{M+1} & = \sum_{n=-M}^{M} h_{n+1,j}(-M-1) k_{nj}(-M-1)
\end{align*}
\]

\[
(9)
\]

\[
\]
These equations can be written as a $2 \times 2$ matrix equation for each of the unknowns $k_{i, j}^{(M+1)}$:

$$
\begin{bmatrix}
1 + k_{i, j}^{(M+1)}(M+1) & \sum_{n=-M+1}^M h_{i, j}^{(M+1)} n k_{n, (M+1)} \\
\sum_{n=-M+1}^M h_{i, j}^{(M+1)} n k_{n, (M+1)} & -1
\end{bmatrix}
\begin{bmatrix}
k_{i, j}^{(M+1)} \\
k_{j, i}^{(M+1)}
\end{bmatrix}
= \begin{bmatrix}
k_{i, j}^{(M+1)} + \sum_{n=-M+1}^M h_{-M+1, n}^{(M+1)} k_{n, (M+1)} \\
\sum_{n=-M+1}^M h_{-M+1, n}^{(M+1)} k_{n, (M+1)}
\end{bmatrix}
$$

where we have used the identities $h_{i, j}^{(M+1)} = g_{i, j}^{(M+1)}$ and $h_{-M+1, n}^{(M+1)} = g_{j, i}^{(M+1)}$. Note that the system matrix in (10) is independent of $i$.

D. Summary of 1-D Algorithm

Given the data $\{y_i\}$ in the interval $[-L, L]$, the entire algorithm for computing the smoothing filters may be summarized as follows:

1) **Initialize** using $g_{i, j}^{0|M} = h_{i, j}$ for all $|i| \leq L$, $1 \leq |j|$.

2) Given $g_{i, j}^{M|M-1}$, update to $g_{i, j}^{M|M}$ as follows:
   a) Compute the boundary points $g_{i, j}^{M|M+1}$ and $g_{i, j}^{M|M-1}$ by solving the $2 \times 2$ system (10).
   b) For each $i$ and $j$, $-M \leq i, j \leq M$, compute $g_{i, j}^{M|M+1}$ using (7). If $g_{i, j}$ has special structure, compute $h_{i, j}$ in parallel using a fast algorithm (e.g., those of [6] or [7]).
   c) Continue for $M = |i| - 1$ to $L$.

III. DERIVATION OF THE 2-D SMOOTHING FILTER ON A POLAR RASTER

A. The Basic Problem

Now we consider the smoothing problem for a two-dimensional random field defined on a polar raster, whose points lie along radial lines in $2N$ angular directions (see Fig. 1). The problem considered is as follows. Given noisy observations $\{y_{i, k}\}$ for all $0 \leq i \leq M, 1 \leq k \leq 2N$ of a zero-mean real-valued discrete random field $\{x_{i, k}\}$ at the points $(i, k)$ of a polar raster on a disk, compute the linear least squares estimate of $x_{i, k}$ for all $(i, k)$ using all of the observations. Here the first subscript denotes radial distance from the origin and the second subscript denotes angular position (as correspondence to the angle $2\pi k / (2N)$).

The observations $\{y_{i, k}\}$ are related to the random field $\{x_{i, k}\}$ by $y_{i, k} = x_{i, k} + n_{i, k}$, where $\{n_{i, k}\}$ is a zero-mean two-dimensional discrete white noise with unit power uncorrelated with $\{x_{i, k}\}$ (white noise with arbitrary power $\sigma^2$ can easily be handled by scaling). The covariance function $k_{i, j; N_1, N_2} \equiv \mathbb{E}[x_{i, k} x_{j, k}]$ of $\{x_{i, k}\}$ is known, and is assumed to be positive semidefinite.

From Fig. 1, it is clear that the point $(i, k) = (-i, k + N)$; in the sequel the point $(i, k)$, $N + 1 \leq k \leq 2N$ will be denoted by $(-i, k - N)$. The linear least squares estimate $\hat{x}_{i, k}$ based on $\{y_{i, k}\}$, $0 \leq j \leq M, 1 \leq k \leq 2N$ is as follows. Given noisy observations $\{y_{i, k}\}$, $-M \leq i \leq M, 1 \leq k \leq 2N$ can be expressed as

$$
\hat{x}_{i, k} = \sum_{j=-M}^M \sum_{j=-N}^N g_{i, j; N_1, N_2} y_{j, k}.
$$

where the smoothing filters $g_{i, j; N_1, N_2} = g_{i, j}^{N_1, N_2}$ satisfy the two-dimensional discrete normal equations

$$
k_{i, j; N_1, N_2} = \sum_{n=-M-1}^M g_{i, j; N_1, N_2} g_{n, (M+1)}
$$

where all $N_1 = 1$.

B. Derivation of the Algorithm

The derivation is identical to that for the one-dimensional case, since the angular sum is unaffected by the increase of the radial sum from $M$ to $M + 1$. The result is (compare to (7)):

$$
\sum_{N_1=1}^N g_{i, j; N_1, N_2} = \sum_{N_1=1}^N g_{i, j; N_1, N_2} - g_{i, j}^{N_1, N_2}
$$

for all $-M \leq i, j \leq M, 1 \leq N_1, N_2 \leq N$. (13)

Here the $h_{i, j; N_1, N_2} \equiv \sum_{N_1=1}^N g_{i, j; N_1, N_2}$ are the two-dimensional prediction filters. The $h_{i, j; N_1, N_2}$ could be computed recursively in parallel with (13), using the fast algorithm of [8].

C. Computation of Boundary Points

As before, we need to compute the boundary points $\sum_{N_1=1}^N g_{i, j; N_1, N_2}$ prior to using (13). Setting $j = \pm(M + 1)$ in (12) results in the equations (compare to (8) and (9))

$$
\begin{align*}
\sum_{N_1=1}^N g_{i, j; N_1, N_2} &= \sum_{N_1=1}^N g_{i, j; N_1, N_2} - \sum_{N_1=1}^N \sum_{N_2=1} g_{i, j; N_1, N_2} - \sum_{N_1=1}^N \sum_{N_2=1} g_{i, j; N_1, N_2}
\end{align*}
$$

for all $-M \leq i, j \leq M, 1 \leq N_1, N_2 \leq N$. (14)
then (14) and (15) can be written in a matrix form as

\[
G^* A + G^* B = R
\]

\[
G^* C + G^* D = S.
\]

Equations (23) and (24) are a \(2N \times 2N\) system of equations for \(G^*\) and \(G^-\); compare them with (10) (for which \(N = 1\)). However, a further simplification is possible. Since the system matrix

\[
\begin{bmatrix}
A & B \\
C & D
\end{bmatrix}
\]

is the same for each \(i\), (23) and (24) can be solved in closed form to give

\[
G^* = (R - SD^{-1}B)(A - CD^{-1}B)^{-1}
\]

\[
G^- = (S - RA^{-1}C)(D - BA^{-1}C)^{-1}
\]

independent of \(i\).

Hence computation of the boundary points \(g_{(M+1),N}^{m+1}\) for all \(i\) requires only the inversion of four \(N \times N\) matrices in (25) and (26). This is significant, since the smoothing filters \(g_{m,N1,N2}\) will generally be computed for all \(i\) and \(N_1, N_2\) (we generally wish to smooth all or most of an image, not just one pixel). This is where our algorithm saves a significant amount of computation, as compared with other algorithms (see below).

D. Summary of 2-D Algorithm

Given the data \(\{y_{i,k} \mid -L \leq i \leq L, 1 \leq k \leq N\}\), the entire algorithm for computing the two-dimensional smoothing filters may be summarized as follows:

1) Initialize using

\[
g_{0,N1,N2} = h_{0,N1,N2}
\]

for all \(-((i-1) \leq j \leq (i+1) \leq N_1, N_2 \leq N\).

2) Given

\[
g_{M,N1,N2}^{m+1}, -M \leq i, j \leq M, 1 \leq N_1, N_2 \leq N,
\]

update to \(g_{M+1,N1,N2}^{m+1}\) follows:

a) Compute the boundary points \(g_{(M+1),N}^{m+1}\) and \(g_{(M+1),N}^{[(M+1),N]}\) by solving in parallel the \(2M + 1 \times 2N\) systems (25) and (26).

b) For each \(i\) and \(j\), \(-M \leq i \leq j \leq M\), and each \(N_1\) and \(N_2\), compute \(g_{(M+1),N}^{m+1}\) from \(g_{(M+1),N}^{m+1}\) using (13). If \(g_{(M+1),N}^{m+1}\) has special structure, compute \(g_{(M+1),N}^{m+1}\) in parallel using a fast algorithm (e.g., the algorithm of [8]).

c) Continue for \(M = |i| - 1 \leq L\).

IV. COMPUTATIONAL COMPLEXITY

We determine the number of multiplications and divisions (MAD's) needed to compute the smoothing filters from the prediction filters. We also determine the total number of MAD's needed to compute the smoothing filters from the covariance function, assuming that the latter has special structure and a fast algorithm has been used to compute the prediction filters. Although some current DSP chips can perform multiplications as quickly as additions, the fact remains that multiplication is a more complex operation than addition. MAD's can still be used as a rough guide to the computational complexity of an algorithm.

A. Computational Complexity of the One-Dimensional Algorithm

The number of MAD's needed to compute the smoothing filters from the prediction filters, given data \(\{y_{i,k} \mid -L \leq j \leq L\}\), can be determined as follows. For each \(i\), updating the smoothing filters from \(g_{m}^{(i)}\) to \(g_{(m+1)}^{(i)}\) (this corresponds to adding two data points at \(j = M + 1\) and \(j = -M + 1\)) requires 6\((2M + 1) + 8\) MAD's to
compute the boundary points $g_{i,j}^{k+M+1}$ (the six sum-of-products computations in (10)), and $2(2M + 1)$ MAD’s to update the other $g_{i,j}^{k+M+1}$ in (7). The total number of MAD’s to compute $g_{i,j}^{k+M+1}$ for one $i$ and all $j$ is thus $\sum_{k=1}^{M} (8(2M + 1) + 8) = 8(L^2 - i^2) + 24L + 8|j| + 16$. However, the total number of MAD’s needed to compute $g_{i,j}^{k+M+1}$ for all $i$ and $j$ is only $\sum_{i=1}^{N} (4(2M + 1) + 2) + \sum_{j=1}^{N} (4(2M + 1) + 6) - (16/3)L^3 + 34L^2 + (146/3)L + 12$, since the system matrix in (10) is independent of $i$, and thus need not be recomputed and reinvected for each $i$.

In the sequel, we assume (for purposes of comparison) that $L >> 1$ and $i >> 1$. Then the dominant terms in the number of MAD’s are the terms of highest order in $L$ and $i$. To facilitate comparisons, only these dominant terms will be given.

If the covariance $k_{ij}$ is Toeplitz, i.e., $\{k_i\}$ is a stationary process, then we will have $k_{i+j} = k_{i-j} = k_{i-j}$ and $g_{i,j}^{M} = g_{i-j}^{M}$ from (2). Then two of the four sum-of-product computations in the system matrix of (10) are redundant, so that computation of $g_{ij}^{M+1}$ for one $i$ and all $j$ requires only $6(L^2 - i^2)$ MAD’s. Also, since $g_{i,j}^{M+1}$ need only be computed for $i \geq 0$, computation of $g_{i,j}^{M+1}$ for all $i$ and $j$ requires only half as many MAD’s as before, viz., $(8/3)L^3$. Furthermore, the Levinson algorithm (L) [6] may be used to compute the prediction filters $h_{i,j}, |L \leq i, j| \leq L$ from $k_{i,j}$, at a cost of $4L^3$ MAD’s. The Levinson algorithm can be propagated in parallel with our algorithm, resulting in an overall fast algorithm for computing the smoothing filters $g_{i,j}^{M+1}$ from $k_{i,j}$. If the covariance is Toeplitz-plus-Hankel, the fast algorithm of [7] may be used to compute the $h_{i,j}, |L \leq i, j| \leq L$ from $k_{i,j}$, at a cost of $24L^2$ MAD’s, again in parallel with our algorithm. However, we no longer have $g_{i,j}^{M+1} = g_{i-j}^{M+1}$, so the reductions in computation for purely Toeplitz covariances no longer apply.

The major alternatives to these procedures are the Levinson-Trench-Zohar (LTZ) [10] algorithm for Toeplitz systems, and the algorithm of Merchant and Parks (MP) [11] for Toeplitz-plus-Hankel systems. We compare the numbers of MAD’s required by all of these algorithms in Table I.

For a Toeplitz covariance, it can be seen from Table I that if $g_{i,j}^{M+1}$ for a single point $i$ is desired, i.e., we wish to compute a smoothed estimate at only one point, then the LTZ algorithm is superior to ours for small values of $i$, while ours is superior for large values of $i$. However, if $g_{i,j}^{M+1}$ for all points $L \leq i \leq L$ is desired, i.e., we wish to compute smoothed estimates at all points (as would generally be the case), then our algorithm in conjunction with the Levinson algorithm requires only $2/3$ as many MAD’s for large $L$. Furthermore, for Toeplitz-plus-Hankel covariances, our algorithm in conjunction with that of [7] requires less than half as many MAD’s to compute $g_{i,j}^{M+1}$ for a single point $i$, and $1/12$ as many MAD’s to compute $g_{i,j}^{M+1}$ for all $i$ when $L$ is large. Further savings are possible since many computations (e.g., the updates and the sum in (10)) can be done in parallel.

Other approaches may require still more computation. $g_{i,j}^{M+1}$ may be updated to $g_{i,j}^{M+1}$ using the well-known formula for updating the inverse of a partitioned matrix. However, this requires $3M^2$ MAD’s per update, as opposed to the $8(2M + 1) + 8$ MAD’s required by the BSK identity. Direct solution of (2) using Gaussian elimination would require $(1/3)(2L + 1)^3 + (1/2)(2L + 1)^2$ MAD’s for each $i$.

### Table I

<table>
<thead>
<tr>
<th>Covariance</th>
<th>Filter for</th>
<th>LTZ or MP L + BSK or [7] + BSK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symmetric</td>
<td>single point $i$</td>
<td>$8L^3$</td>
</tr>
<tr>
<td>Toeplitz</td>
<td>all $</td>
<td>L \leq i \leq L$</td>
</tr>
<tr>
<td>Toeplitz-plus</td>
<td>single point $i$</td>
<td>$64L^3$</td>
</tr>
<tr>
<td>Hankel</td>
<td>all $</td>
<td>L \leq i \leq L$</td>
</tr>
</tbody>
</table>

N × N matrix multiplications and inversions for (25) and (26).

Updating the other smoothing filters from $B_{i,j}^{k+M+1}$ to $g_{i,j}^{k+M+1}$ requires $2(2M + 1)N^3$ MAD’s for (13). Hence the number of MAD’s needed to compute $g_{i,j}^{k+M+1}$ from the prediction filters for one $i$ and $j$ is $8(L^2 - i^2)N^3$, while the number of MAD’s needed for all $i$ and $j$ is $(16/3)L^3N^3$. Note that these are the numbers for the one-dimensional algorithm multiplied by $N$, since all operations now involve matrices.

In the sequel, we assume (for purpose of comparison) that $L >> N >> 1$. If the covariance $k_{i,j}$ is block Toeplitz, i.e., Toeplitz in $i$ and $j$, then the Levinson-Wiggins-Robinson (LWR) [12] algorithm may be used to compute the prediction filters $h_{i,j}$ from $k_{i,j}$, at a cost of $6L^3N^3$ MAD’s (recall that the backward predictors are no longer the time-reversed forward predictors in the multichannel case). If the covariance is Toeplitz-plus-Hankel in both $i$ and $j$ and $N_i$ and $N_j$, as it is for an isotropic random field on a polar raster, the fast algorithm of [8] may be used to compute the prediction filters, at a cost of $24L^2N^2$ MAD’s.

The major alternatives to these procedures are the LWR algorithm adapted to an arbitrary block-Toeplitz system, and a matrix generalization of the Merchants-Parks procedure for block Toeplitz-plus-Hankel systems. Results are summarized in Table II. The savings are similar to those for the one-dimensional algorithms, except for the even greater savings for block Toeplitz-plus-Hankel covariances. The reason for the great savings here is the efficiency of the algorithm of [8], which requires only $24L^2$ MAD’s to determine the prediction filters from the covariance function; that is negligible compared to $8(L^2 - i^2)N^3$ and $(16/3)L^3N^3$ if $L >> N >> 1$.

### C. Relation to Continuous-Parameter BSK Identities

It is instructive to examine the continuous-parameter limits of the various equations of this correspondence. Let the intervals between points be $\delta_i$ in the radial direction and $\delta_i = 2\pi/N$ rad in the angular direction. Introducing a radial weighting factor, as discussed below (12), and taking limits as $\delta_i$ and $\delta_j$ go to zero results in the following transformations:

1. The discrete normal equations (2) and (12) become Fredholm integral equations. Similarly, the discrete Wiener-Hopf equation (3) and its two-dimensional counterpart become Wiener-Hopf integral equations.

2. The smoothing filters become the Fredholm resolvents to the integral operators associated with the covariance functions.

3. Using $g_{i,j}^{M+1} = g_{i-j}^{M+1}, h_{i,j}^{M+1}$, and equation (7) becomes

$$
\frac{\partial g}{\partial T}(x, y; T) = -(g(x, T; h(T, y) + g(x, T; -h(T, y)))
$$

where $g(x, y; T)$ is the smoothing function by which an observation at $y$ in the interval $[-T, T]$ is multiplied and integrated to compute an estimate at $x$. Equation (27) is the BSK resolvent identity (mod-
where $e_n$, $e'$, and $s$ are unit vectors, $x = |x| e_n$, $y = |y| e_n$, and $S$ is the unit circle. Equation (28) is identical to the generalized BSK identity applied to a multidimensional continuous-parameter smoothing problem in [9];  

4) Similarly, the recursion (13) becomes  

$$  \frac{\partial}{\partial T} \left( g(x, y|Te', T)h(Te', y|e')T de' \right) = 0 $$  

(28)  

where $e_n$, $e'$, and $e'$ are unit vectors, $x = |x| e_n$, $y = |y| e_n$, and $S$ is the unit circle. Equation (28) is identical to the generalized BSK identity applied to a multidimensional continuous-parameter smoothing problem in [9];  

5) Since $\delta_k$ becomes a continuous-time impulse, the units in (10) and (19), (20) dominate the other terms. Hence the computations of the boundary points (10) and (25), (26) become, respectively,  

$$  g(x, y|Te', T)h(Te', y|e')T de' $$  

(28)  

where $e_n$, $e'$, and $e'$ are unit vectors, $x = |x| e_n$, $y = |y| e_n$, and $S$ is the unit circle. Equation (28) is identical to the generalized BSK identity applied to a multidimensional continuous-parameter smoothing problem in [9];  

Note that although the discrete equations transform into the expected continuous equations, the forms of the discrete equations are not obvious from the continuous equations.  

V. CONCLUSION  

New fast algorithms for computing the linear least squares smoothing filters for random processes and fields have been derived. These algorithms relate the smoothing filters to the prediction filters associated with the same covariance. If the covariance has special structure, fast algorithms such as those of [6]-[8] may be used to compute the prediction filters; such algorithms may be propagated in parallel with those of this correspondence. This can result in significant computational savings. However, it is important to emphasize that the results of this correspondence hold for arbitrary covariances, and do not rely on the existence of such fast algorithms.  

In the limit of continuous time, the one-dimensional algorithm reduces to the BSK identity, which was applied previously to smoothing problems for continuous-time stationary random processes by Kailath. However, the algorithms are nontrivial discrete and two-dimensional generalizations of the BSK identity. Since both data and numerical computation are inherently discrete in nature, these algorithms constitute a significant step in the practical application of these smoothing ideas.

### TABLE II

<table>
<thead>
<tr>
<th>Covariance</th>
<th>Filter for</th>
<th>LWR or MP</th>
<th>LWR + BSK or [8] + BSK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Toeplitz</td>
<td>single point $i$</td>
<td>$10L^2N^3$</td>
<td>$(14L^2 - 4i)N^3$</td>
</tr>
<tr>
<td></td>
<td>all $-L \leq i \leq L$</td>
<td>$8L^2N^3$</td>
<td>$(16/3)L^2N^3$</td>
</tr>
<tr>
<td>Block-Toeplitz plus Hankel</td>
<td>single point $i$</td>
<td>$64L^2N^3$</td>
<td>$(8L^2 - 4i)N^3$</td>
</tr>
<tr>
<td></td>
<td>all $-L \leq i \leq L$</td>
<td>$64L^2N^3$</td>
<td>$(16/3)L^2N^3$</td>
</tr>
</tbody>
</table>

Corrections to "Comments on 'A Curious Concerning Discrete Time Convolution'," plus a Remark

Anibal R. Figueiras-Vidal

In my comments1 to a correspondence by Hall and Wise [1], a typographic error appeared in the discussion below (7a, b): "iff $B_i \neq \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots $ should be "iff $B_i = r \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots "$.

Also, it should be pointed out that, although the example proposed for getting an identically zero convolution of two nowhere zero sequences was not formed by $L_i$ sequences, if $(\sin (B_i)n)/\pi n$ is changed to $(\sin (B_i)n)/\pi n)^k \cos \Omega_n \rightarrow 0$, then the sequences become $L_i$ norm sequences in these cases (8) should be changed to $\Omega_i + kB_n/2 < B_i - kB_n/2$.  

### REFERENCES


