PARALLEL SMOOTHING[1]

A.H. Tewfik[2], A.S. Willsky[3], and B.C. Levy[4]

ABSTRACT

This paper describes a smoothing algorithm that involves the parallel processing of the data in subintervals with little communication among the processors. Data in the subintervals are first processed in parallel starting from subinterval centers and processing outward to the subinterval boundaries. After an exchange of information among processors, a final set of parallel recursions, proceeding inward in each subinterval, yields the desired estimates. The proposed procedure is found in general to have a total on–line computational complexity slightly higher than that of the non–parallel implementations. However, since several processors are used in parallel, the running time of this algorithm is much smaller than that of a single smoother solution. Furthermore when the process to be estimated is reversible, an even–odd decomposition of the process yields a block diagonalization that yields a further, considerable reduction in the required computations.

---

[1]This work was performed at MIT and was supported in part by the National Science Foundation under Grant ECS–8700903 and in part by the Army Research Office under Grant DAAL03–86–K–0171.

[2]Department of Electrical Engineering, Univ. of Minnesota, Minneapolis, MN 55455.

[3]Laboratory for Information and Decision Systems and Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.

[4]Department of Electrical Engineering, Univ. of California, Davis, CA 95616.

I. Introduction

The advent of cheap and powerful processors in the past few years, together with the relatively high cost of communication has made decentralized estimation schemes extremely attractive and has led to the development of a number of parallel processing algorithms for optimal smoothing for linear state variable models. In this paper we present a new algorithm of this type which is highly parallel in nature and requires minimal communication among processors. As in [1]–[4], our algorithm involves the partitioning of the data interval of interest into subintervals, processing all data segments in parallel and then combining the results of these local processing steps. However, the approach we present is a significantly different alternative to these earlier methods. To understand our approach conceptually, it is useful to review two of the standard approaches to smoothing illustrated in Figure 1. One of these is the Mayne–Fraser two–filter smoother [5] in which the smoothed estimate is computed by a forward–filtered estimate and a reverse–filtered estimate. These two estimates can be computed in parallel, resulting in a total run time proportional to twice the length of the entire data interval (once for the filter computations, the other for their combination) and requiring the storage of these two intermediate estimates over the entire interval. Note also that both processors must have access to *all* of the data. A second approach, the Rauch–Tung–Striebel algorithm [5], begins with the forward filtered estimate. At the end of this processing step we have the full smoothed estimate at the terminal point, and we then proceed with a reverse recursion for the smoothed estimate, using the forward filtered estimate as input. Because of its serial nature the total run time of this algorithm is proportional to twice the data interval length but requires the storage of only one intermediate estimate over the entire interval.

It is relatively straightforward to devise an algorithm that represents a modest improvement to the computational demands of these two algorithms. Specifically, as illustrated in Figure 1(a), we divide the data interval in half. The first processing step then consists of the parallel implementation of a forward and reverse filter each of which operates over only one half of the data interval. At the point at which these computations meet, we can combine the estimates as in the Mayne–Fraser smoother to obtain the smoothed estimate at that point, which

can then be used to initialize parallel Rauch–Tung–Striebel recursions on each subinterval. In this case the total processing time is proportional to the overall data interval (we have two parallel Rauch–Tung–Striebel algorithms over half the interval length) plus one addithonal calculation at the centerpoint. Data storage consists of two filtered estimates, but each over only half of the data interval. Note also that each of the processors needs to access only half of the data and the communications between processors is limited to the very simple trading of filtered estimates at the common interval endpoint.

In this paper we present the generalization of this simple algorithm to finer subdivisions of the data interval. As we will see, this can lead to significant reductions in processing time and data accessing requirements, with minimal interprocessor communication. When we divide the interval into three or more pieces, the question arises as to directions in which recursions should proceed in each subinterval. The algorithm we describe in Section III involves recursions that propagate *radially outward toward* and *inward from* the boundary points of each subinterval. The key to developing these recursions is the use of a joint dynamic model, described in the next section, for a stochastic process x(t) and its time–reversed version x(–t). As we will also see in Section II, considerable simplifications arise for the case of a stationary, time–reversible process if we transform the joint x(t), x(–t) dynamics into a form yielding the even–odd decomposition of the process. A second question concerns the generalization of the Mayne–Fraser combining step when we need to merge information at *several* boundary points. As we describe in Section III, this generalization consists of a discrete two–filter–like computation involving only the interval end points. The result is a parallel procedure which has several attractive features and which is especially efficient for time–reversible processes.

II. Outward Dynamic Models and Even–Odd Decompositions

Consider the following, n–dimensional dynamic model defined for $-T/2 \le t \le T/2$

$$\dot{x}(t) = F(t)x(t) + G(t)u(t) \tag{2.1}$$

$$y(t) = H(t)x(t) + v(t) \tag{2.2}$$

where u(t) and v(t) are independent, zero–mean white noise processes with unit intensity, independent of the initial condition x(–T/2). From [6] we have that the reversed–time Markovian model for x(t) is given by

$$-\dot{x}(t) = -[F(t)+G(t)G^T(t)\Pi^{-1}(t)]x(t) - G(t)\tilde{u}(t) \tag{2.3}$$

where $\Pi(t)$ is the covariance of x(t) satisfying

$$\dot{\Pi}(t) = F(t)\Pi(t) + \Pi(t)F^T(t) + G(t)G^T(t) \tag{2.4}$$

and $\tilde{u}(t)$ is a unit intensity white noise independent of the future of x(t). If we define

$$z(t) = \Pi^{-1}(-t)x(-t) \quad , \quad w(t) = \tilde{u}(-t) \tag{2.5}$$

some algebra yields

$$\dot{z}(t) = F^T(-t)z(t) - \Pi^{-1}(-t)G(-t)w(t) \tag{2.6}$$

and combining this with (2.1), (2.2) yields the following 2n–dimensional dynamic model defined for $0 \le t \le T/2$:

$$\begin{bmatrix}\dot{x}(t)\\ \dot{z}(t)\end{bmatrix} = \begin{bmatrix}F(t) & 0\\ 0 & F^T(-t)\end{bmatrix}\begin{bmatrix}x(t)\\ z(t)\end{bmatrix} + \begin{bmatrix}G(t) & 0\\ 0 & -\Pi^{-1}(-t)G(-t)\end{bmatrix}\begin{bmatrix}u(t)\\ w(t)\end{bmatrix} \tag{2.7}$$

$$\begin{bmatrix}y(t)\\ y(-t)\end{bmatrix} = \begin{bmatrix}H(t) & 0\\ 0 & H(t)\Pi(-t)\end{bmatrix}\begin{bmatrix}x(t)\\ z(t)\end{bmatrix} + \begin{bmatrix}v(t)\\ v(-t)\end{bmatrix} \tag{2.8}$$

where, by construction, $[u^T(t)\ ,\ w^T(t)]^T$ and $[v^T(t)\ ,\ v^T(-t)]^T$ are independent, unit intensity white noise processes. Note that (2.7), (2.8) describes a dynamic system for the joint evolution of x(t) and x(–t), propagating outward from 0 to ±T/2. Note also that while (2.7), (2.8) appear to describe decoupled evolution for the two parts of the state, statistical coupling exists, thanks to the initial condition [x(0), z(0)] which has as its (singular covariance)

$$\begin{bmatrix}\Pi(0) & I\\ I & \Pi^{-1}(0)\end{bmatrix}$$

To carry our analysis further, we focus on the time invariant stationary case, i.e. when F,G, and H are constant and x(t) is a stationary process. We also assume that (F,G,H) is a

minimal realization so that, in particular, the constant state covariance matrix Π is the unique positive definite solution of the Lyapunov equation

$$F\Pi + \Pi F^T + GG^T = 0 \tag{2.9}$$

Also, without loss of generality, we can assume that G has full column rank and, thanks to the following result, that Π is diagonal and that there exists a signature matrix $S = \text{diag}\,(I_{n_1}, -I_{n_2})$, $n_1 + n_2 = n$, such that

$$SF = F^T S \tag{2.10}$$

Proposition: A minimal model of the form (2.1)–(2.2) with F,G,H constant and Π satisfying (2.9) can be transformed into another minimal realization $\{\bar{F}, \bar{G}, \bar{H}, \bar{\Pi}\}$ such that there exists a signature matrix S obeying

$$S\bar{F} = \bar{F}^T S \ ,$$

and such that $\bar{\Pi}$ is diagonal.

Proof: First, use the transformation

$$\bar{\bar{x}}(t) = \Pi^{-1/2} x(t)$$

to obtain a new realization $\{\bar{\bar{F}}, \bar{\bar{G}}, \bar{\bar{H}}, I\}$ in which the variance of the state process is identity. Next, find a symmetric nonsingular P such that[5]

$$P\bar{\bar{F}} = \bar{\bar{F}}^T P \tag{2.11}$$

Decompose P as

$$P = V\Lambda^{1/2} S \Lambda^{1/2} V^T$$

where V is a nonsingular orthogonal matrix, $\Lambda^{1/2}$ is diagonal, and S is a signature matrix. Note that $\Lambda^{1/2} S \Lambda^{1/2} = S\Lambda$ has the eigenvalues of P on its diagonal. Apply the transformation

---

[5]The existence of a possibly nonsymmetric P (satisfying (2.11)) follows from the similarity of $\bar{F}$ and $\bar{F}^T$. However in this case, P^T has the same property, as does the symmetric matrix $(P+P^T)$ (see also [11]).

$$\bar{x}(t) = \Lambda^{1/2} V^T \bar{\bar{x}}(t)$$

to obtain a new minimal realization $\{\bar{F}, \bar{G}, \bar{H}, \Lambda\}$. It is a simple matter to check that the state variance

$$E[\bar{x}(t)\bar{x}^T(t)] = \Lambda$$

is diagonal, and that

$$S\bar{F} = \bar{F}^T S.$$

■■

Assume now that Π is dagonal and that S and F satisfy (2.10), and consider the following change of variables, yielding the *even* and *odd* state processes respectively:

$$x_e(t) = x(t) + Sz(t) \tag{2.13}$$

$$x_o(t) = x(t) - Sz(t) \tag{2.14}$$

and the even and odd observations $y_e(t)$ and $y_o(t)$ as

$$y_e(t) = y(t) + y(-t) \tag{2.15}$$

$$y_o(t) = y(t) - y(-t) \tag{2.16}$$

From (2.7), (2.8), specialized to the stationary case, and (2.10) we find that

$$\begin{bmatrix} \dot{x}_e(t) \\ \dot{x}_o(t) \end{bmatrix} = \begin{bmatrix} F & 0 \\ 0 & F \end{bmatrix} \begin{bmatrix} x_e(t) \\ x_o(t) \end{bmatrix} + \begin{bmatrix} G & -\Pi^{-1}SG \\ G & \Pi^{-1}SG \end{bmatrix} \begin{bmatrix} u(t) \\ w(t) \end{bmatrix} \tag{2.17}$$

with

$$E\left[\begin{bmatrix} x_e(0) \\ x_0(0) \end{bmatrix} [x_e^T(0)\ x_0^T(0)] \right] = \begin{bmatrix} (I+\Pi^{-1}S)(\Pi+S) & \Pi-\Pi^{-1} \\ \Pi-\Pi^{-1} & (I-\Pi^{-1}S)(\Pi-S) \end{bmatrix} \tag{2.18}$$

and

$$\begin{bmatrix} y_e(t) \\ y_o(t) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} H(I+\Pi S) & H(I-\Pi S) \\ H(I-\Pi S) & H(I+\Pi S) \end{bmatrix} \begin{bmatrix} x_e(t) \\ x_o(t) \end{bmatrix} + \begin{bmatrix} I & I \\ I & -I \end{bmatrix} \begin{bmatrix} v(t) \\ v(-t) \end{bmatrix} \tag{2.19}$$

In general there is no particular reason to prefer the model (2.17)–(2.19) over the model (2.7), (2.8). However, we do obtain a considerable simplification if the process y(t) is

statistically time reversible [7], i.e. if $E[y(t)y^T(0)]$ is symmetric for all t (note that this is always true if y(t) is scalar). In this case the results of [7] imply that

$$\Pi = I \quad , \quad HS = H \quad , \quad SG = GQ \tag{2.20}$$

where Q is an orthogonal matrix. From this we find that (2.17) reduces to

$$\begin{bmatrix} \dot{x}_e(t) \\ x_o(t) \end{bmatrix} = \begin{bmatrix} F & 0 \\ 0 & F \end{bmatrix} \begin{bmatrix} x_e(t) & 0 \\ 0 & x_o(t) \end{bmatrix} + \sqrt{2} \begin{bmatrix} G & 0 \\ 0 & G \end{bmatrix} \begin{bmatrix} \mu_1(t) \\ \mu_2(t) \end{bmatrix} \tag{2.21}$$

where $\mu_1(t)$ and $\mu_2(t)$ are independent white noise processes of unit intensity. Also the initial covariance for (2.21) is

$$4 \text{ diag } (I_{n_1}, \ 0, \ 0, \ I_{n_2})$$

so that $x_e(t)$ and $x_o(t)$ are independent processes (note that the initial uncertainty in x(0) is distributed between $x_e(0)$ and $x_o(0)$ according to the structure of S). Furthermore (2.19) becomes

$$\begin{bmatrix} y_e(t) \\ y_o(t) \end{bmatrix} = \begin{bmatrix} H & 0 \\ 0 & H \end{bmatrix} \begin{bmatrix} x_e(t) \\ x_o(t) \end{bmatrix} + \sqrt{2} \begin{bmatrix} \xi_1(t) \\ \xi_2(t) \end{bmatrix} \tag{2.22}$$

where $\xi_1(t)$ and $\xi_2(t)$ are independent white noise processes with unit intensity, so that the even and odd measurements are decoupled as well.

III The Parallel Smoothing Algorithm

In this section we describe a parallel algorithm for computing the optimal estimate of x(t) satisfyingy (2.1) for $T_o \le t_1 \le T_N$ given the measurements (2.2) over the same interval. The procedure we describe involves three steps. To begin, we divide $[T_0, T_N]$ into N equal intervals $[T_{i-1}, T_i]$, i = 1,..., N, each of length T.

Step 1: As illustrated in Figure 2(a) this step consists of the parallel computation of outward filtered estimates on each subinterval. Specifically, consider one of these intervals, say the ith, and let $-T/2 \le t \le T/2$ denote the independent variable measured relative to the center of this interval, namely $T_{i-1} + T/2$. Over this interval, we recursively compute $\hat{x}(t|-t,t)$ and $\hat{x}(-t|-t,t)$

and their error covariances for $0 \le t \le T/2$, where $\hat{x}(s|-t,t)$ denotes the estimate of $x(s)$ based on $\{y(\tau) \mid |\tau| \le t\}$. Using the similarity transformations as described in the preceding section, we see that this is a standard Kalman filtering computation using, for example, the model (2.7), (2.8) for a general, time varying model or (2.21), (2.22) for stationary models with time–reversible outputs.

Step 2: Let us now revert back to the original time reference. From the endpoints of the subinterval computations of Step 1 we now have computed $\hat{x}(T_{i-1}|T_{i-1},T_i)$ and $\hat{x}(T_i|T_{i-1},T_i)$, i.e. the estimates of the endpoints given local data, and their corresponding error covariances $P(T_{i-1}|T_{i-1},T_i)$ and $P(T_i|T_{i-1},T_i)$. What we accomplish during the second step of the computation is to use these local estimates to compute $\hat{x}(T_i|T_o,T_N)$, i.e. that full smoothed estimate at each of the endpoints based on *all* of the data. The form of the required calculations, which, can be deduced from the smoothing results of [8], [9], consists of a Mayne–Fraser–like two–filter computation involving the endpoints only, as illustrated in Figure 2(b). In particular the forward recursion computes the estimates $\hat{x}(T_{i-1}|T_o,T_i)$ and $\hat{x}(T_i|T_o,T_i)$ and the corresponding error covariances as i increases, starting with initial conditions $\hat{x}(T_o|T_o,T_1)$ and $\hat{x}(T_1|T_o,T_1)$. The processing involves communication from processor to processor as i increases, using only the endpoint filtered estimates computed in Step 1.

Specifically using the results of [8] and [9], we obtain the following recursions:

$$\hat{x}(T_{i-1}|T_0,T_i) = P(T_{i-1}|T_0,T_i)\,[P^{-1}(T_{i-1}|T_0,T_{i-1})\,\hat{x}(T_{i-1}|T_0,T_{i-1}) + P^{-1}(T_{i-1}|T_{i-1},T_i)\,\hat{x}(T_{i-1}|T_{i-1},T_i)] \tag{3.1}$$

$$\hat{x}(T_i|T_0,T_i) = \hat{x}(T_i|T_{i-1},T_i) + \Phi_0^i(-T/2,T/2)\,[\hat{x}(T_{i-1}|T_0,T_i) - \hat{x}(T_{i-1}|T_{i-1},T_i)] \tag{3.2}$$

The covariances $P(T_{i-1}|T_{i-1},T_i)$ is one of the endpoint covariances computed in Step 1. The other covariances and quantities required in (3.1), (3.2) are computed recursively as follows

(again based on [8], [9]):

$$P^{-1}(T_{i-1}|T_0,T_i) = P^{-1}(T_{i-1}|T_0,T_{i-1}) + P^{-1}(T_{i-1}|T_{i-1},T_i) - \Pi^{-1} \tag{3.3}$$

$$P(T_i|T_0,T_i) = P(T_i|T_{i-1},T_i) + \Phi_0^i(-T/2,T/2)\,[P(T_{i-1}|T_0,T_i) - P(T_{i-1}|T_{i-1},T_i)]\Phi_0^{iT}(-T/2,T/2) \tag{3.4}$$

where the transition matrix Φ_0^i for each interval is calculated as follows, where we again revert to the locally centered variable t:

$$\dot{\Phi}_0^i(-T/2,t) = (F(t) - P_0^i(t)\,H^T(t)H(t))\Phi_0^i(-T/2,t) \tag{3.5}$$

$$\Phi_0^i(-T/2,-T/2) = I \tag{3.6}$$

Where $P_0^i(t)$ is computed from

$$\dot{P}_0^i(t) = F(t)P_0^i(t) + P_0^i(t)F^T(t) + G(t)G^T(t) - P_0^i(t)H^T(t)H(t)P_0^i(t) \tag{3.7}$$

$$P_0^i(-T/2) = 0 \tag{3.8}$$

Alternatively, $\Phi_0(-T/2,T/2)$ can be obtained as

$$\Phi_0^i(-T/2,T/2) = P(T_{i-1},T_i)\,P^{-1}(T_{i-1}|T_{i-1},T_i) \tag{3.9}$$

(c.f [9]), where $P(T_{i-1},T_i)$ is the cross–covariance of the step 1 filtering errors at $t = T_{i-1}$ and $t = T_i$, and is readily obtained from the step 1 covariance calculations. The term Π^{-1} is subtracted in eq. (3.3) to account for the fact that the a priori information on x(•) was used twice, in computing both $\hat{x}(T_{i-1}|T_0,T_{i-1})$ and $\hat{x}(T_{i-1}|T_{i-1},T_i)$.

In parallel with this forward recursion, there is also an analogous backward recursion. Specifically, a set of equations for computing $\hat{x}(T_i|T_i,T_N)$ and $P(T_i|T_i,T_N)$, given $\hat{x}(T_{i+1}|T_{i+1},T_N)$ and $P(T_{i+1}|T_{i+1},T_N)$ can easily be derived from [8], [9], and in fact, this set of equations is very similar to the set of equations (3.3) – (3.6).

Note that, at the end of this calculation, $\hat{x}(T_i|T_0,T_i)$, $\hat{x}(T_i|T_i,T_N)$ and their respective covariances $P(T_i|T_0,T_i)$ and $P(T_i|T_i,T_N)$ for all i are available, and can be used to compute the

optimal smoothed estimates of x(t) at all of the endpoints, using standard smoothed results:

$$\hat{x}(T_i|T_0,T_N) = P_s(T_i|T_0,T_N)\,(P^{-1}(T_i|T_0,T_i)\,\hat{x}(T_i|T_0,T_i) + P^{-1}(T_i|T_i,T_N)\,\hat{x}(T_i|T_i,T_N)) \tag{3.10}$$

$$P_s^{-1}(T_i|T_0,T_N) = P^{-1}(T_i|T_0,T_i) + P^{-1}(T_i|T_i,T_N) - \Pi^{-1} \tag{3.11}$$

Step 3: In this last step, the data is processed in parallel in a radially inward direction toward the center of each interval, to yield the optimal smoothed estimate of x(t) for all t. Let us again revert to the locally centered time index for the ith interval. The computation (3.10), (3.11) then provides us with the optimal smoothed estimate of $x(\cdot)$ at the endpoints. As illustrated in Figure 2(c), we can then use the Rauch–Tung–Striebel algorithm (based, for example, on the model (2.7), (2.8) or (2.21), (2.22)) starting from these endpoints in order to reprocess the filtered estimate in an inward direction in order to compute the optimal smoothed estimate. Specifically let $x_{eo}(t)$ denote the state of our outward model (i.e. as in (2.7) or (2.21)). Then the optimal smoothed estimate of $x_{eo}(t)$, $\hat{x}^s_{eo}(t)$, for $0 \le t \le T/2$ is obtained as the solution of the backwards equation

$$\frac{d\hat{x}^s_{eo}(t)}{dt} = (\mathcal{F} + \mathcal{G}\mathcal{G}^T P_{eo}^{-1}(t))\, x^s_{eo}(t) - \mathcal{G}\mathcal{G}^T P_{eo}^{-1}(t)\, \hat{x}_{eo}(t) \tag{3.12}$$

with the initial condition $\hat{x}^s_{eo}(T/2)$ obtained from Step 2. Here, the matrices $\mathcal{F}$ and $\mathcal{G}$ are the dynamic matrices of the outward model (from (2.7) or (2.21)) and $\hat{x}_{eo}(t)$ and $P_{eo}(t)$ are the filtered estimate and error covariance calculated in Step 1.

IV Computational Complexity

Let us first focus on the on–line complexity, both in terms of total computations required and the efficiencies due to parallelization. A careful examination of Steps 1 and 3 of our algorithm reveals that the total computational load, in the worst case, is roughly 4/3 times the total load of

the standard Rauch–Tung–Striebel algorithm. Since the actual run time of these two steps is proportional to 1/N times this load, we see that substantial savings in run time are achievable if a number of processors are used. Furthermore, in the reversible case the total load of Steps 1 and 3 *equals* that of Rauch–Tung–Striebel, yielding a further savings. Of course these savings are somewhat countered by the on–line computations involved in Step 2. Note that Step 2 only involves updating estimates at the interval endpoints, which, unless N is quite large, are quite few in comparison to the total number of data points. Thus the on–line load of Step 2 is typically negligible compared to that of Steps 1 and 3. It is worth noting, however, that the total run time for our algorithm is the sum of a term proportional to 1/N (Steps 1 and 3) and a term proportional to N (Step 2), so that there is an optimum number of processors in terms of minimizing run time. Note also that our algorithm offers advantages in data accessing, as each processor needs to use only a small part of the data, and the cost of this is the communication of n numbers (the forward and backward recursions of Step 2) to each of its neighbors. Note also that the total computational complexity of our procedure is lower than that of other parallel algorithms.

Finally, let us briefly comment on the off–line complexity. In general the off–line computational requirements for Steps 1 and 3 are roughly twice those for the Rauch–Tung–Striebel algorithm, while in the reversible case the complexity for these steps is the same as for the standard algorithm. The off–line computations involved in Step 2 (given by (3.3), (3.4), (3.9), and (3.11)) are comparatively expensive per point, but again there are usually relatively few such endpoints. Furthermore for stationary processes (3.9) need only be calculated once.

V. Conclusion

In this paper we presented a new parallel smoothing algorithm based on a partitioning of the data interval and the use of outward dynamic models in each subinterval, leading to parallel outward–recursive processing in each interval, followed by the propagation of information concerning interval endpoints and then parallel inward–recursive processing. The total on–line

computational complexity of this procedure is at worst only marginally higher than that of non–parallel implementations. However, since a number of parallel processors are used, the running time of this algorithm is much smaller than that of single smoother procedures. A natural extension of this work is to consider parallel algorithms for smoothing for boundary–value processes—i.e. processes described locally by a model of the form (2.1) – (2.2) but with noncausal boundary conditions (c.f. [10]). An interesting issue is the interpretation of information contained in data outside a particular interval as a boundary measurement. With such an interpretation, we should be able to use the results of Adams et al. [10] to derive another class of parallel smoothing algorithms. Furthermore it should also be possible to extend these ideas to estimation for two–dimensional fields, and in this case the savings in run time and in data accessing should be even more dramatic.

References

[1] U. B. Desai and B. Das, "Parallel Algorithms for Kalman Filtering," Proceedings of the 1985 American Control Conference, Boston, MA, June 1985, pp. 920–921.

[2] S.R. McReynolds, "Parallel Filtering and Smoothing Algorithms," IEEE Trans. Auto. Control., Vol. AC–19, pp. 556–561, 1974.

[3] M. Morf, J.R. Dobbins, B. Friedlander, and T. Kailath, "Square–root Algorithms for Parallel Processing in Optimal Estimation," Automatica, Vol. 15, pp. 299–306, 1979.

[4] G. Meyer and H. Weinert, "An Approach to Reliable Parallel Data Processing," Proc. of the 21st IEEE Conf. on Decision and Control, Orlando, Florida, 1982.

[5] A. Gelb, ed., Applied Optimal Estimation, The M.I.T. Press, Cambridge, MA, 1974.

[6] O. Taussky and H. Zassenhaus, "On the Similarity Transformation between a Matrix and its Transpose," Pacific J. Math., Vol. 9, pp. 893–896, 1959.

[7] J. C. Willems, "Time Reversibility in Deterministic and Stochastic Dynamical Systems," Proc. 1977 US–Italy Seminar on Variable Structure Systems, R. R. Mohler and A. Ruberti Eds., Springer–Valag, Berlin, New York, 1978, pp. 318–326.

[8] D.G. Lainiotis, "Optimal Linear Smoothing; Continous Data Case," Int. J. Contr., Vol. 17, pp. 921–930, 1973.

[9] G. Verghese, B. Friedlander, and T. Kailath, "Scattering Theory and Linear Least–Squares Estimation, Part III: The Estimates," IEEE Trans. on Autom. Control, Vol. AC–25, pp. 794–802, 1980.

[10] M.B. Adams, A.S. Willsky, and B.C. Levy, "Linear Estimation of Boundary Value Stochastic Processes, Part II; 1–D Smoothing Problems," IEEE Trans. Autom. Control, Vol. AC–29, No. 9, pp. 811–821, 1984.

[11] O. Taussky and H. Zassenhaus, "On the Similarity Transformation between a Matrix and Its Transpose," Pacific J. Math., Vol. 9, pp. 893–896, 1959.

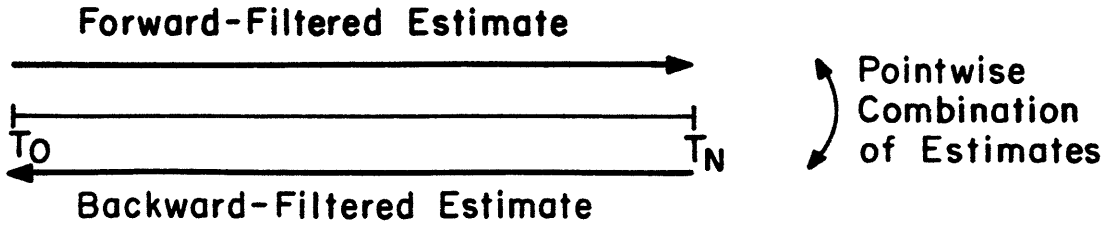
Figure Captions

Fig. 1 Illustrating Three Processing Structures for Optimal Smoothing

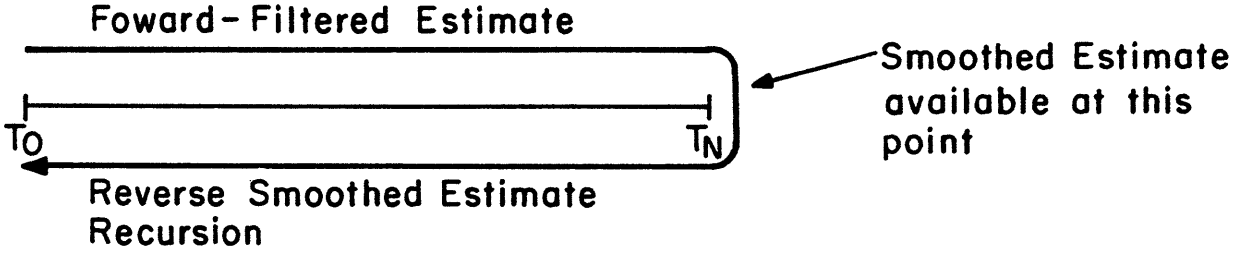
(a) Mayne–Fraser Processing Structure
(b) Rauch–Tung–Striebel Processing Structure
(c) A Simple Parallel Algorithm

Fig. 2 Parallel processing algorithm:

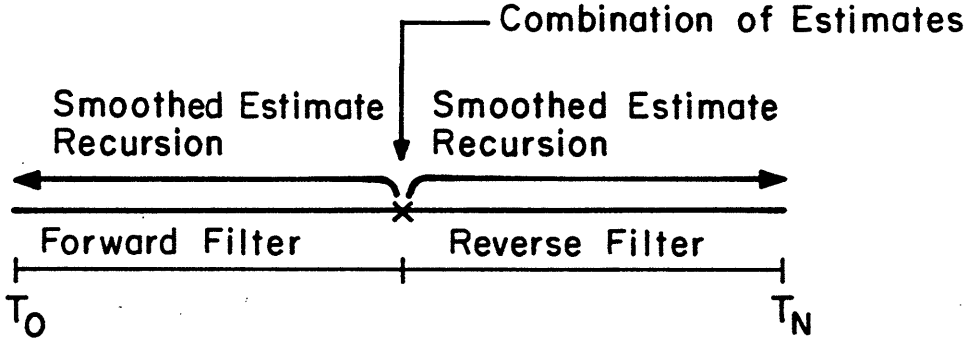
(a) Step 1: Outward filter propagation
(b) Step 2: Communication among processors to estimate endpoints
(c) Step 3: Inward computation of smoothed estimate


Forward-Filtered Estimate
Pointwise
Combination
of Estimates
T_O
T_N
Backward-Filtered Estimate

(a)

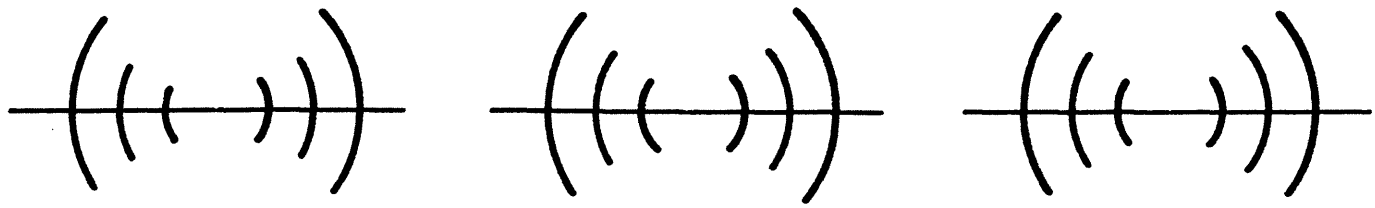

Foward-Filtered Estimate
Smoothed Estimate
available at this
point
T_O
T_N
Reverse Smoothed Estimate
Recursion

(b)


Combination of Estimates
Smoothed Estimate
Recursion
Smoothed Estimate
Recursion
Forward Filter
Reverse Filter
T_O
T_N

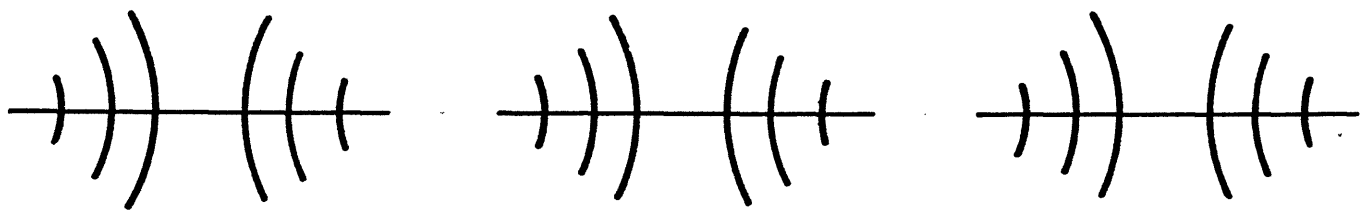
(c)

Figure 1



(a)

(b)



(c)

Figure 2