ML Parameter Estimation of a Multiscale Stochastic Process Using the EM Algorithm

Ashvin Kannan, Mari Ostendorf, William C. Karl, David A. Castañon, and Randall K. Fish

***Abstract*—An algorithm for estimation of the parameters of a multiscale stochastic process based on scale-recursive dynamics on trees is presented. The expectation-maximization algorithm is used to provide maximum likelihood estimates for the general case of a nonhomogeneous tree with no fixed structure for the process dynamics. Experimental results are presented using synthetic data.**

***Index Terms*—Maximum likelihood parameter estimation, multiscale process.**

I. Introduction

Multiscale stochastic processes represent an important class of models, of which a particularly useful subclass is based on scale-recursive dynamics on trees [1], [2]. These models allow efficient algorithms for both estimation and likelihood calculation [1], [3], resulting in a variety of applications, e.g., [2]–[4]. Denoting a node in the tree by t with parent $t\bar{\gamma}$, a state-space model for the evolution in scale of the Gaussian tree-based process X and its noisy observation Y can be written as

$$x(t) = A(t)x(t\bar{\gamma}) + w(t) \tag{1}$$

$$y(t) = C(t)x(t) + v(t) \tag{2}$$

where $x(t)$ is the state of the process at node t. The root node state $x(0)$ has distribution $\mathcal{N}(0, \Sigma(0))$, where $\mathcal{N}(\mu, \Sigma)$ denotes a Gaussian density with mean μ and covariance Σ. The process noise $w(t)$ is independent and identically distributed (or "white"), independent of $x(0)$, and has distribution $\mathcal{N}(0, Q(t))$. The state $x(t)$ is observed via a noisy measurement $y(t)$, where the measurement noise $v(t)$ is white, independent of $x(0)$ and $w(t)$, and has distribution $\mathcal{N}(0, R(t))$. Note that the zero-mean assumptions of the root node $x(0)$ and the noise terms are not a requirement of the model but result in simpler estimation equations. Nonzero mean problems can be solved by subtracting out the mean, solving the resulting zero-mean problem, and then adding back the mean.

Relatively little work has been done in parameter estimation of multiscale processes of the form of (1) and (2) from data. The few existing algorithms assume a restricted model structure [5], [6] or use *ad hoc* fitting methods [4]. It is reported that the expectation-maximization (EM) algorithm has been used for maximum-likelihood (ML) parameter estimation for a binary tree [7], [8], but no details were provided, and the solution is apparently different from that described here [9].

In this correspondence, we provide the solution to the problem of ML estimation of the parameters of a nonhomogeneous tree from observation of multiple independent runs for the general case with nonuniform (or irregular) branching and where no fixed structure is assumed for the dynamics of the process, i.e., $A(t), Q(t), C(t)$, and $R(t)$ can vary from node to node. An example of such a problem is in speech recognition in the context of adaptation of acoustic models to a new speaker [10]. Our application of the EM algorithm for ML parameter estimation follows the basic approach of [11] and builds on results from [1] and [12].

Manuscript received August 18, 1998; revised November 30, 1999. The work was performed when all the authors were at Boston University and was supported in part by the Office of Naval Research under Grant N00014-92-J-1778, the Air Force Office of Scientific Research under Grant F49620-96-1-0028, and the Army Research Office under Grant ARO DAAG55-97-1-0013. The associate editor coordinating the review of this paper and approving it for publication was Prof. Arnab K. Shaw.

A. Kannan is with Nuance Communications, Menlo Park, CA 94025 USA.

M. Ostendorf is with University of Washington, Seattle, WA 98195 USA.

W. C. Karl, D. A. Castañon, and R. K. Fish are with Boston University, Boston, MA 02215 USA.

Publisher Item Identifier S 1053-587X(00)04071-X.

II. RTS Smoothing for a Multiscale Tree Process

In this section, we list equations to compute smoothed estimates of the tree state $x(t)$, which will also be needed in parameter estimation. Define that $Y_t = \{y(\sigma) \,|\, \sigma = t \text{ or } \sigma \text{ is a descendant of } t\}$ and that $Y_t^+ = \{y(\sigma) \,|\, \sigma \text{ is a descendant of } t\}$. For a set of measurements Y, define $\hat{x}(t \,|\, Y) \triangleq E\{x(t) \,|\, Y\}$ and the associated error covariance $P(t \,|\, Y) \triangleq E\{[x(t) - \hat{x}(t \,|\, Y)][x(t) - \hat{x}(t \,|\, Y)]^T\}$. Given Y_0, i.e., *all* the observations, smoothed estimates of the state $\hat{x}(t \,|\, Y_0)$ and error covariance $P(t \,|\, Y_0)$ can be computed using a generalization of the Rauch–Tung–Striebel (RTS) algorithm, as in [1] for binary trees and given here for the general case of a nonhomogeneous tree.

Smoothing is performed in two sweeps: an upward sweep starting at the leaves to compute filtered estimates $\hat{x}(t \,|\, Y_t)$ and $P(t \,|\, Y_t)$ followed by a downward sweep starting at the root to compute smoothed estimates $\hat{x}(t \,|\, Y_0)$ and $P(t \,|\, Y_0)$. For notational convenience, we abbreviate the Y dependence by using $\hat{x}(t \,|\, t)$ for $\hat{x}(t \,|\, Y_t)$, $\hat{x}(t \,|\, t+)$ for $\hat{x}(t \,|\, Y_t^+)$, and $\hat{x}_s(t)$ for $\hat{x}(t \,|\, Y_0)$, and similarly for P.

Upward Sweep. The root node covariance $\Sigma(0)$ is propagated to all nodes by the Lyapunov equation $\Sigma(t) = A(t)\Sigma(t\bar{\gamma})A^T(t) + Q(t)$. For a node t, $\Sigma(t)$ is the unconditional error covariance, implied by the root node covariance $\Sigma(0)$, whereas $P(t \,|\, Y)$ is the error covariance conditioned on the measurements Y. The upward sweep starts at the leaves with the initialization $\hat{x}(t \,|\, t+) = 0$ and $P(t \,|\, t+) = \Sigma(t)$. Suppose we have $\hat{x}(t \,|\, t+)$ and $P(t \,|\, t+)$ at node t, where we update these estimates by incorporating the measurement $y(t)$ as in [1]:

$$\hat{x}(t \,|\, t) = \hat{x}(t \,|\, t+) + K(t)[y(t) - C(t)\hat{x}(t \,|\, t+)] \tag{3}$$

$$P(t \,|\, t) = [I - K(t)C(t)]P(t \,|\, t+) \tag{4}$$

where

$$K(t) = P(t \,|\, t+)C^T(t)[C(t)P(t \,|\, t+)C^T(t) + R(t)]^{-1}. \tag{5}$$

The upward sweep uses a model for the dynamics

$$x(t\bar{\gamma}) = F(t)x(t) + \bar{w}(t)$$

$F(t) = \Sigma(t\bar{\gamma})A^T(t)\Sigma^{-1}(t)$ and $E\{\bar{w}(t)\bar{w}^T(t)\} = \bar{Q}(t) \triangleq \Sigma(t\bar{\gamma}) - \Sigma(t\bar{\gamma})A^T(t)\Sigma^{-1}(t)A(t)\Sigma(t\bar{\gamma})$. For the general case of a nonhomogeneous tree, let the $q(t)$ children of node t be denoted by $t\alpha_1 \ldots t\alpha_{q(t)}$. With $\hat{x}(t\alpha_i \,|\, t\alpha_i)$, we can predict $x(t)$ from each child and find the associated error covariance by

$$\hat{x}(t \,|\, t\alpha_i) = F(t\alpha_i)\hat{x}(t\alpha_i \,|\, t\alpha_i) \tag{6}$$

$$P(t \,|\, t\alpha_i) = F(t\alpha_i)P(t\alpha_i \,|\, t\alpha_i)F^T(t\alpha_i) + \bar{Q}(t\alpha_i). \tag{7}$$

The different estimates of $x(t)$ predicted by its children are then merged to get

$$\hat{x}(t \,|\, t+) = P(t \,|\, t+)\sum_{i=1}^{q(t)} P^{-1}(t \,|\, t\alpha_i)\hat{x}(t \,|\, t\alpha_i) \tag{8}$$

$$P(t \,|\, t+) = \left[(1 - q(t))\Sigma^{-1}(t) + \sum_{i=1}^{q(t)} P^{-1}(t \,|\, t\alpha_i)\right]^{-1}. \tag{9}$$

The upward sweep proceeds until we reach the root node computing $\hat{x}_s(0) = \hat{x}(0\,|\,0)$ and $P_s(0) = P(0\,|\,0)$.

Downward Sweep. The estimates at the root node $\hat{x}_s(0)$ and $P_s(0)$ serve as the starting point for the downward recursion, which follows as in [1]

$$\hat{x}_s(t) = \hat{x}(t\,|\,t) + J(t)\,[\hat{x}_s(t\bar{\gamma}) - \hat{x}(t\bar{\gamma}\,|\,t)] \tag{10}$$

$$P_s(t) = P(t\,|\,t) + J(t)\,[P_s(t\bar{\gamma}) - P(t\bar{\gamma}\,|\,t)]\,J^T(t) \tag{11}$$

$$J(t) = P(t\,|\,t)F^T(t)P^{-1}(t\bar{\gamma}\,|\,t). \tag{12}$$

There is an alternative form of the RTS smoothing algorithm, where $\Sigma^{-1}(t)$ is set to zero at all nodes during the upward sweep, and the prior of the root node is added between the upward and downward sweeps. See [13] and [14] for more information. Either approach can be used in the EM estimation framework.

III. EM Algorithm for Tree Parameter Estimation

We take the approach proposed in [11] for ML parameter estimation of a stochastic linear system from multiple independent runs using the EM algorithm [15], which is known to converge but possibly to a local (i.e., not ML) optimum.[1] Let the parameters of the multiscale process $\Sigma(0)$ and $\{A(t), Q(t), C(t), R(t)\,|\,t \in \mathcal{T}\}$, where $\mathcal{T}$ is the set of all the nodes in the tree, be collectively denoted by θ. The state and observation dimensions are assumed to be known (i.e., not part of θ) but may vary with scale. For the ith run, let X^i denote all the states in the tree, and let Y_0^i denote all the observations. The EM algorithm iteratively maximizes the overall expected log-likelihood of the observed data and the missing data (all states and missing observations). For independent runs $i = 1, \ldots, N$, the expected log-likelihood is

$$\sum_{i=1}^{N} E_\theta\{\mathcal{L}(X^i, Y_0^i, \theta)\,|\,Y_0^i\} \tag{13}$$

where

$$\begin{aligned}\mathcal{L}(X, Y_0, \theta) = &- \sum_{t\in(\mathcal{T}-0)} \{\log|Q(t)| + [x(t) \\ &- A(t)x(t\bar{\gamma})]^T Q^{-1}(t)[x(t) - A(t)x(t\bar{\gamma})]\} \\ &- \sum_{t\in\mathcal{T}} \{\log|R(t)| + [y(t) - C(t)x(t)]^T \\ &\times R^{-1}(t)[y(t) - C(t)x(t)]\} + \text{constant}.\end{aligned} \tag{14}$$

Each iteration involves two steps: 1) the expectation or E-step, where conditional expectations of complete-data sufficient statistics are computed; and 2) the maximization or M-step, where these are used in re-estimating model parameters. Below, we summarize each step, beginning with the M-step to show what quantities will be needed in the E-step.

A. M-Step

Define the operator $\langle\ \rangle$ to represent the average of smoothed estimates generated by each of the N runs, e.g.,

$$\langle g(x,t)\rangle = \frac{1}{N}\sum_{i=1}^{N} E\{g(x,t)|Y_0^i\}.$$

[1]In practice, the problem of local optima can be addressed by using multiple start points. In addition, in our experience with various EM estimation problems in speech processing applications, convergence typically requires only a few iterations.

These averages can be thought of as "expected" sufficient statistics of the parameters. Maximizing the expected likelihood using multivariable regression to obtain new estimates of tree parameters gives

$$\hat{A}(t) = \langle x(t)x^T(t\bar{\gamma})\rangle\,\langle x(t\bar{\gamma})x^T(t\bar{\gamma})\rangle^{-1} \tag{15}$$

$$\hat{Q}(t) = \langle x(t)x^T(t)\rangle - \hat{A}(t)\langle x(t\bar{\gamma})x^T(t)\rangle \tag{16}$$

$$\hat{C}(t) = \langle y(t)x^T(t)\rangle\,\langle x(t)x^T(t)\rangle^{-1} \tag{17}$$

$$\hat{R}(t) = \langle y(t)y^T(t)\rangle - \hat{C}(t)\langle x(t)y^T(t)\rangle \tag{18}$$

$$\hat{\Sigma}(0) = \langle x(0)x^T(0)\rangle. \tag{19}$$

Note that $\hat{Q}(t), \hat{R}(t)$ and $\hat{\Sigma}(0)$ should be symmetric positive semi-definite matrices. If structure is imposed on the process, e.g., if a parameter is node-invariant or scale-variant (but constant in a scale), the domain of the averaging operator is increased to include all the nodes $t \in \mathcal{T}_s$ in the subset $\mathcal{T}_s$ that share parameters, i.e.,

$$\langle g(x)\rangle = \frac{1}{N|\mathcal{T}_s|}\sum_{i=1}^{N}\sum_{t\in\mathcal{T}_s} E\{g(x,t)\,|\,Y_0^i\}. \tag{20}$$

B. E-Step

The E-step of the algorithm computes the expected quantities for each run required in the right hand side of (15)–(19). For a run Y_0^i, these quantities can be written in terms of the observations, when present, and smoothed estimates of the states and their associated error covariances:

$$E\{x(t)\,|\,Y_0^i\} = \hat{x}_s(t) \tag{21}$$

$$E\{x(t)x^T(t)\,|\,Y_0^i\} = P_s(t) + \hat{x}_s(t)\hat{x}_s^T(t) \tag{22}$$

$$E\{x(t)x^T(t\bar{\gamma})\,|\,Y_0^i\} = P_s(t,t\bar{\gamma}) + \hat{x}_s(t)\hat{x}_s^T(t\bar{\gamma}) \tag{23}$$

where $P_s(t,t\bar{\gamma}) \triangleq E\{[x(t) - \hat{x}_s(t)][x(t\bar{\gamma}) - \hat{x}_s(t\bar{\gamma})]^T\,|\,Y_0^i\}$. The terms $\hat{x}_s(t)$ and $P_s(t)$ are computed by the downward sweep of the RTS algorithm. The remaining term required is $P_s(t,t\bar{\gamma})$, which we show can also be computed using terms from the RTS downward sweep. When observations at some nodes are missing, the missing observations and the unseen state information are jointly treated as missing data by the EM algorithm. For each run Y_0^i, for missing $y(t)$

$$E\{y(t)\,|\,Y_0^i\} = C(t)E\{x(t)\,|\,Y_0^i\} \tag{24}$$

$$E\{y(t)y^T(t)\,|\,Y_0^i\} = R(t) + C(t)E\{x(t)x^T(t)\,|\,Y_0^i\}C^T(t) \tag{25}$$

$$E\{y(t)x^T(t)\,|\,Y_0^i\} = C(t)E\{x(t)x^T(t)\,|\,Y_0^i\}. \tag{26}$$

The smoothed error $\tilde{x}_s(t) \triangleq x(t) - \hat{x}_s(t)$ has been shown in [12] to be modeled as a multiscale process of the following form:

$$\tilde{x}_s(t) = J(t)\tilde{x}_s(t\bar{\gamma}) + \breve{w}(t) \tag{27}$$

where $\breve{w}(t)$ is zero mean and white with covariance given by $P(t\,|\,t) - P(t\,|\,t)F^T(t)P^{-1}(t\bar{\gamma}\,|\,t)F(t)P(t\,|\,t)$ and is independent of $\{\tilde{x}_s(\sigma)\,|\,\sigma$ is neither t nor a descendant of $t\}$. Using (27), we can compute

$$\begin{aligned}P_s(t,t\bar{\gamma}) &= E\{\tilde{x}_s(t)\tilde{x}_s^T(t\bar{\gamma})\,|\,Y_0^i\} \\ &= E\{[J(t)\tilde{x}_s(t\bar{\gamma}) + \breve{w}(t)]\tilde{x}_s^T(t\bar{\gamma})\,|\,Y_0^i\} \\ &= J(t)P_s(t\bar{\gamma}).\end{aligned} \tag{28}$$

We now have all the quantities needed for (21)–(23) (the E-step) and, in turn, (15)–(19) (the M-step). Performing iterations of these two steps will result in an estimate of θ that is generally referred to as an ML estimate, even though the stopping point may be only a local maximum in likelihood.

A degenerate tree with only one leaf node (all parent nodes have only one child) can be interpreted as a standard linear dynamical system,

i.e., having a time-like index (note that the process state $x(t)$ is still a vector in general). For this case, the above algorithm gives the *same* results as that in [11]; however, they differ in the manner of computation of $P_s(t, t\bar{\gamma})$. In [11], filtering proceeds from the root to the leaf computing $P(t, t\bar{\gamma} \mid t)$ followed by smoothing from leaf to root to compute $P_s(t, t\bar{\gamma})$ using a modified RTS algorithm. In our case, filtering proceeds from the leaf to the root, followed by smoothing from the root to the leaf using the regular RTS algorithm. We compute $P_s(t, t\bar{\gamma})$ directly in the downward sweep by using the result that the smoothed error is a Gauss–Markov process.

IV. Experimental Results

Experiments with synthetic data were conducted to investigate convergence of the algorithm and accuracy of the parameter estimation under various conditions. Data (the y terms) were generated for the systems according to (1) and (2). To restrict the scope of the experiments, we considered only tree processes with node-invariant parameters, i.e., $A(t) = A, Q(t) = Q$, and $R(t) = R$. The matrix $C(t) = I$ (identity) is constrained for identifiability. In most experiments, the EM iterations estimated $\Sigma(0), A$ and Q, using R held constant at its true value, similar to our end application in speech recognition [10]. Trees with both scalar and vector observation spaces were explored, using fixed dimensions independent of scale and a balanced binary tree with 32 leaves (total of 63 nodes). These simple examples were chosen to illustrate convergence behavior of the algorithm and performance degradation associated with having observations only at the leaves. Detailed examination of the algorithm is problem dependent and left for other work.

A. Scalar State and Observation Spaces

For the scalar case to compare estimated and actual parameters, we define a normalized error $\text{err}_A = \text{abs}(A - \hat{A})/\text{abs}(A)$, where $\text{abs}(A)$ is the absolute value of A. Fig. 1 shows the error associated with the different parameter estimates at each EM iteration for the initialization given in Table I and 100 independent runs of the process. The top plot is for the case when all observations are used and indicates rapid convergence (5–10 iterations). The bottom plot shows results when only observations at the leaves are used, where convergence is much slower (roughly 30 or more iterations). When observations are only at the leaves, the $\Sigma(0)$ estimate initially diverges and does not start converging toward the true value until the A estimate is close to the true value, leading to behavior that is often not monotonic, as illustrated in the figure. Table I gives the parameter estimates for these two conditions at 50 iterations and 200 iterations, showing that accurate estimates are obtained. The difference in convergence rates for the cases with and without observations at internal nodes is even more pronounced when the system is "unstable." Increasing A from 0.9 to 2.0 had little effect when data were available at all nodes. However, with data only at the leaves, $\Sigma(0)$ was still two orders of magnitude from convergence after 300 iterations.

Several other experimental configurations were run to investigate behavior of the estimation algorithm. Performance is not very sensitive to the values used for parameter initialization; therefore, the estimation error is probably not due to local optima. Initializing an order of magnitude above and below the true parameter values resulted in similar convergence rates and parameter estimates even for the leaf-only observation case. Changing the number of runs from ten to 500 showed little or no impact on system performance. Finally, when R is unknown and estimated, convergence slowed significantly for the case where observations are only at the leaves, as we might expect. Systems with observations at all nodes had similar convergence rates and final estimates as for the case when R is known.

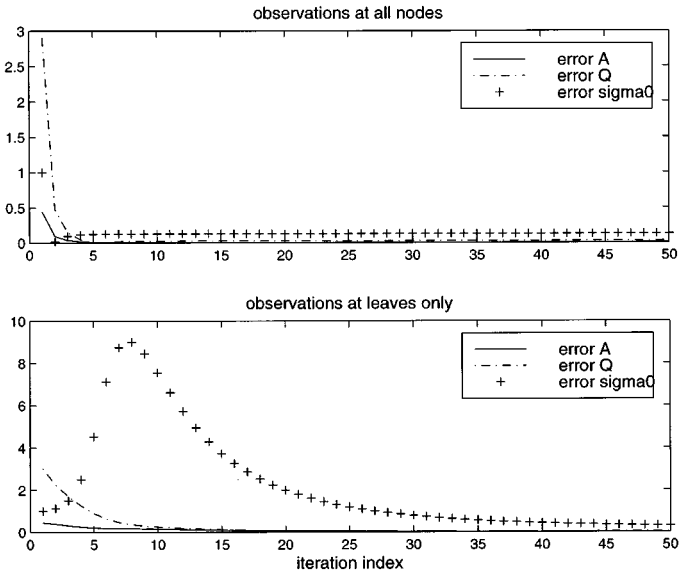

Fig. 1. Parameter estimation error for $\Sigma(0), A$, and Q in a system with scalar state and observation spaces (6-levels) as a function of the number of EM iterations. The top plot shows the case where observations are available at all nodes, and the bottom shows the case where observations are only at the leaf nodes. Actual values are $\Sigma(0) = 1.0, A = 0.9, Q = 0.5$.

TABLE I
Parameters for System with Scalar State and Observation Spaces: Initial Estimates, Values After Specified Number of Iterations for Cases Where Observations Are at All Nodes versus Only at the Leaves, and True Values. $C = 1.0$ and $R = 0.1$ Are Fixed and Known

Parameter estimates	$\Sigma(0)$	A	Q
Initial	2.000	0.500	2.000
All nodes – 50 iter	0.873	0.896	0.487
All nodes – 200 iter	0.873	0.896	0.487
Leaves only – 50 iter	1.306	0.890	0.480
Leaves only – 200 iter	1.217	0.894	0.476
True	1.000	0.900	0.500

B. Vector State and Observation Spaces

For the vector case, 400 runs of 2-D y vectors were generated. The larger number of runs was used since the number of free parameters is greater in the vector case than the scalar case. The true system parameters are shown in Table II, which also shows the initial values for the EM iterations and the estimated values after 200 iterations. To compare estimated and actual matrix parameters, we define a (scalar) normalized error $\text{err}_A = ||A - \hat{A}||_F / ||A||_F$, where $||A||_F \triangleq (\sum_{ij} |A_{ij}|^2)^{(1/2)}$ is the Frobenius norm. The normalized error as a function of EM iteration is shown in Fig. 2 for the first 100 iterations. The top two plots are for the case when observations at all nodes are used, whereas the bottom two plots use observations only at the leaves. The plots again indicate that convergence is slower when observations are only available at the leaf nodes versus at all nodes, particularly for the root node covariance $\Sigma(0)$. The difference in convergence rates is also more pronounced than for the scalar case. Experiments with different system parameters, including diagonal matrices, show similar behavior, sometimes with more oscillations in the $\Sigma(0)$ estimation error than in the scalar case when the observations are only at the leaves, as illustrated in Fig. 3. The oscillations are likely due to the fact that there is ambiguity

TABLE II
PARAMETERS FOR SYSTEM WITH VECTOR STATE AND OBSERVATION SPACES WITH $C = I$: INITIAL ESTIMATES, VALUES AFTER SPECIFIED NUMBER OF ITERATIONS FOR CASES WHERE OBSERVATIONS ARE AT ALL NODES VERSUS ONLY AT THE LEAVES, AND TRUE VALUES

Parameter estimates	$\Sigma(0)$	A	Q	R
Initial	$\begin{bmatrix} 2.0 & 0.0 \\ 0.0 & 4.0 \end{bmatrix}$	$\begin{bmatrix} 0.5 & 0.0 \\ 0.0 & 0.3 \end{bmatrix}$	$\begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$	fixed to true
All nodes – 200 iter	$\begin{bmatrix} 1.027 & 1.034 \\ 1.034 & 2.186 \end{bmatrix}$	$\begin{bmatrix} 0.893 & 0.111 \\ 0.399 & 0.602 \end{bmatrix}$	$\begin{bmatrix} 0.504 & 0.105 \\ 0.105 & 0.505 \end{bmatrix}$	fixed to true
Leaves only – 200 iter	$\begin{bmatrix} 1.483 & 0.312 \\ 0.312 & 2.449 \end{bmatrix}$	$\begin{bmatrix} 0.934 & 0.046 \\ 0.451 & 0.541 \end{bmatrix}$	$\begin{bmatrix} 0.495 & 0.104 \\ 0.104 & 0.513 \end{bmatrix}$	fixed to true
True	$\begin{bmatrix} 1.0 & 0.9 \\ 0.9 & 2.0 \end{bmatrix}$	$\begin{bmatrix} 0.9 & 0.1 \\ 0.4 & 0.6 \end{bmatrix}$	$\begin{bmatrix} 0.5 & 0.1 \\ 0.1 & 0.5 \end{bmatrix}$	$\begin{bmatrix} 0.1 & 0.05 \\ 0.05 & 0.1 \end{bmatrix}$

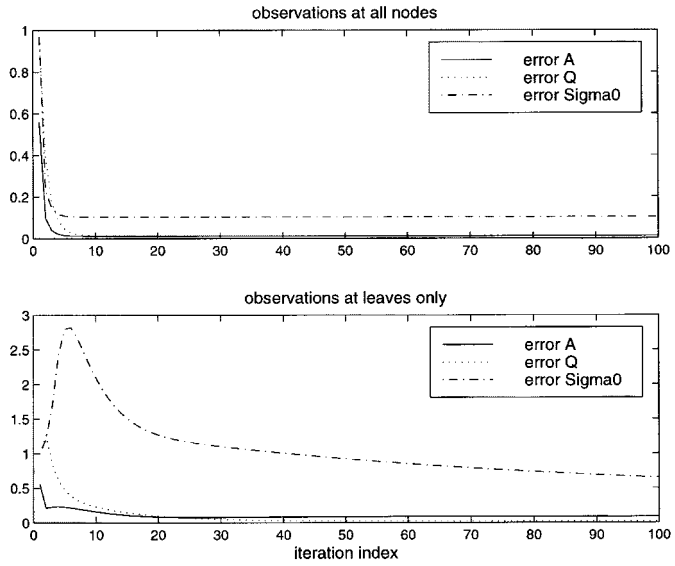

Fig. 2. Parameter estimation error (using the Frobenius norm) for $\Sigma(0)$, A, and Q in a system with vector state and observation spaces (6-levels) as a function of the number of EM iterations. The top plot shows the case where observations are available at all nodes, and the bottom shows the case where observations are only at the leaf nodes.

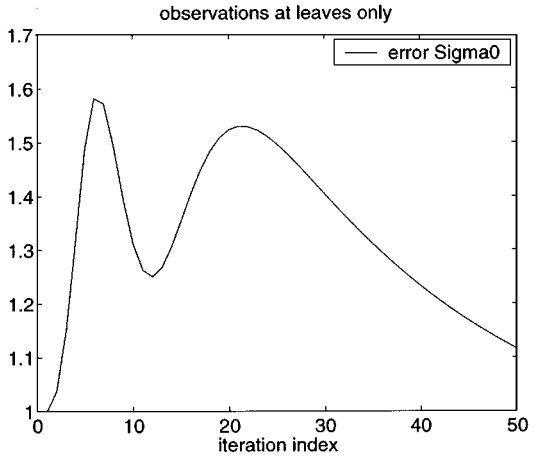

Fig. 3. Parameter estimation error (using the Frobenius norm) for $\Sigma(0)$ in a system with vector state and observation spaces (6-levels) as a function of the number of EM iterations. This example illustrates oscillation of the error and is for a case where the true $\Sigma(0)$ is diagonal.

in the error between $\Sigma(0)$, A, and Q, all of which account for variations in the observations at the leaf nodes. Overestimates of A result in underestimates of Q and $\Sigma(0)$, and vice versa. These types of oscillations are not surprising in EM-type algorithms, which are fixed-point iterations rather than gradient descent. When observations are available at all nodes, the estimation of A, Q, and $\Sigma(0)$ is more decoupled.

V. DISCUSSION

In summary, we have presented an ML estimator for the parameters of multiscale stochastic processes based on scale-recursive dynamics on trees. This estimator provides a rational method for finding the parameters of such models for general tree structures, in contrast to the *ad hoc* approaches that have been used to date. In this correspondence, we have only presented the ML estimator and demonstrated its operation. Further work is needed to completely characterize its behavior and performance in a variety of application domains.

The computational complexity of the tree RTS smoother is $O(d^3 n)$, where d is the dimensionality of the state (assuming a fixed dimension), and n is the number of nodes (internal plus leaf) in the tree. The memory requirements are $O(d^2 n)$, due to the need to store covariances. The algorithm is inherently parallelizable, although our implementation was on serial machines. Since the E-step of the EM algorithm for parameter estimation uses the RTS smoother, the overall complexity of training is $O(d^3 nrp)$, where r is the number of runs of the multiscale process, and p is the number of EM iterations. The computational needs of the M-step are insignificant in comparison with the E-step.

REFERENCES

[1] K. C. Chou, A. S. Willsky, and A. Benveniste, "Multiscale recursive estimation, data fusion, and regularization," *IEEE Trans. Automat. Contr.*, vol. 39, pp. 464–478, Mar. 1994.

[2] M. R. Luettgen, W. C. Karl, A. S. Willsky, and R. R. Tenney, "Multiscale representations of Markov random fields," *IEEE Trans. Signal Processing*, vol. 41, pp. 3377–3396, Dec. 1993.

[3] M. R. Luettgen and A. S. Willsky, "Likelihood calculation for a class of multiscale stochastic models, with application to texture discrimination," *IEEE Trans. Image Processing*, vol. 4, pp. 194–207, Feb. 1995.

[4] P. Fieguth, W. C. Karl, A. S. Willsky, and C. Wunsch, "Multiresolution optimal interpolation and statistical analysis of Topex/Poseidon satellite altimetry," *IEEE Trans. Geosci. Remote Sensing*, vol. 33, pp. 280–292, Apr. 1995.

[5] P. Fieguth and A. S. Willsky, "Fractal estimation using models on multiscale trees," *IEEE Trans. Signal Processing*, vol. 44, pp. 1297–1300, May 1996.

[6] V. V. Digalakis and K. C. Chou, "Maximum likelihood identification of multiscale stochastic models using the wavelet transform and the EM algorithm," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, vol. IV, Minneapolis, MN, Apr. 1993, pp. 93–96.

[7] K. C. Chou and L. P. Heck, "A multiscale stochastic modeling approach to the monitoring of mechanical systems," in *Proc. IEEE-SP Int. Symp. Time-Freq. Time-Scale Anal.*, Philadelphia, PA, Oct. 1994, pp. 25–27.
[8] K. C. Chou, "Maximum-likelihood estimation of multiscale stochastic model parameters," presented at the Proc. of the IEEE-SP Int. Symp. Time-Freq. Time-Scale Anal., Paris, France, June 1996.
[9] ——, personal communication, unpublished, Sept. 1996.
[10] A. Kannan and M. Ostendorf, "Modeling dependency in adaptation of acoustic models using multiscale tree processes," in *Proc. Eur. Conf. Speech Commun. Technol.*, vol. 4, 1997, pp. 1863–1866.
[11] V. Digalakis, J. Rohlicek, and M. Ostendorf, "ML estimation of a stochastic linear system with the EM algorithm and its application to speech recognition," *IEEE Trans. Speech Audio Processing*, pp. 431–442, Oct. 1993.
[12] M. R. Luettgen and A. S. Willsky, "Multiscale smoothing error models," *IEEE Trans. Automat. Contr.*, vol. 40, pp. 173–175, Jan. 1995.
[13] K. C. Chou, A. S. Willsky, and R. Nikoukhah, "Multiscale systems, Kalman filters, and Ricatti equations," *IEEE Trans. Automat. Contr.*, vol. 39, pp. 479–492, Mar. 1994.
[14] A. Kannan, "Adaptation of spectral trajectory models for large vocabulary continuous speech recognition," Ph.D. dissertation, Boston Univ., Boston, MA, 1997.
[15] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood estimation from incomplete data," *J. R. Stat. Soc. B*, vol. 39, no. 1, pp. 1–38, 1977.

Constant Geometry Algorithm for Discrete Cosine Transform

Jarmo Takala, David Akopian, Jaakko Astola, and Jukka Saarinen

***Abstract*—Modification to the architecture-oriented fast algorithm for discrete cosine transform of type II from Astola and Akopian is presented, which results in a constant geometry algorithm with simplified parameterized node structure. Although the proposed algorithm does not reach the theoretical lower bound for the number of multiplications, the algorithm possesses the regular structure of the Cooley–Tukey FFT algorithms. Therefore, the FFT implementation principles can also be applied to the discrete cosine transform.**

***Index Terms*—Fast transforms, perfect shuffle, regular algorithms.**

I. Introduction

The objective for developing fast algorithms for discrete trigonometric transforms is typically minimization of the number of arithmetic operations, especially the number of multiplications. However, the number of operations is not the only cost measure when realizing the transform algorithms. This can be illustrated by the fact that Winograd's Fourier transform algorithm [2], although it has the smallest number of multiplications in the known fast Fourier transform (FFT) algorithms, is rarely used in practice due to its overall complexity [3]. However, several FFT implementations based on the regular Cooley–Tukey FFT algorithms [4] have been reported. Therefore, the regularity in the topology of the signal flow graph of the algorithm represents an important cost measure, especially in parallel hardware implementations. In this sense, the proposed fast algorithms for discrete sine and cosine transforms (DST and DCT, respectively) are expensive since, in general, they do not possess the regular form of the Cooley–Tukey FFT.

Recently, regular algorithms have also been suggested for these transforms, e.g., in [1] and [5]. The derivation of these algorithms is based on localizing the structural irregularities of the algorithm into nodes of the Cooley–Tukey FFT-type computational graph. The resulting algorithms contain similar data reorderings between the computational stages, i.e., the interconnection topology is constant, hence, the name constant geometry algorithm. Such regular algorithms lend themselves for several linear mapping methods, as illustrated with the Cooley–Tukey FFT algorithms. An interesting alternative for parallel implementations is a partial column architecture where the operations in the signal flow graph of the algorithm can be mapped onto fewer and, more importantly, a varying number of processing elements. Therefore, the performance of such a scalable architecture can be tailored according to given throughput requirements by changing the number of processing elements, i.e., the area can be traded against the speed, as described in [1] and [5].

The architecture-oriented fast algorithms for DCT and DST in [1] are represented as consecutive stages of computations consisting of set of nodes with four inputs and four outputs. The nodes are described in unified manner, where the operations of a node are defined by parameters dependent on the position of the particular node in the signal flow graph of the algorithm. Although the algorithms in [1] are constant geometry, one of the computational stages has a specific form: Two parameterized nodes are connected in cascade describing a single node in the constant geometry structure. In other words, the number of arithmetic operations in the nodes at different stages varies, and therefore, specific scheduling of the operations is required for recursive implementations. This is inconvenient, especially when the forward and inverse algorithms are implemented on the same architecture.

In this correspondence, we present a modification to the architecture-oriented algorithms in [1], which results in a) constant geometry algorithms where the operations are evenly distributed over the node functions and b) simplified parametrized node structure. The modification is only shown for the DCT of type II algorithm, but the same procedure can also be applied to other regular DCT and DST algorithms proposed in [1]. The main idea is in the observation that the algorithms in [1] could be equivalently interpreted as computationally balanced but not constant geometry.

II. Preliminaries

The *perfect shuffle* permutation reorders an N-point data sequence $X = (x_0, x_1, \cdots, x_{N-1})^T$ to another sequence $Y = (x_{ps_N(0)}, x_{ps_N(1)}, \cdots, x_{ps_N(N-1)})^T$ according to a function $ps_N(\cdot)$ defined as

$$ps_N(i) = \begin{cases} i/2, & i \text{ is even} \\ (i+N-1)/2, & i \text{ is odd.} \end{cases} \quad (1)$$

The *perfect unshuffle* is the inverse permutation for perfect shuffle. Here, perfect shuffle and unshuffle permutation matrices of order N are denoted by $P^T_{N,2}$ and $P_{N,2}$, respectively. These permutations can

Manuscript received April 19, 1999; revised December 11, 1999. This work was supported by Nokia Foundation. The associate editor coordinating the review of this paper and approving it for publication was Prof. Chaitali Chakrabarti.

J. Takala and J. Saarinen are with the Digital and Computer Systems Laboratory, Tampere University of Technology, Tampere, Finland.
D. Akopian is with Nokia Mobile Phones, Tampere, Finland.
J. Astola is with the Signal Processing Laboratory, Tampere University of Technology, Tampere, Finland.

Publisher Item Identifier S 1053-587X(00)04053-8.