# Fast Implementation of the Continuous Wavelet Transform with Integer Scales

Michael Unser, Akram Aldroubi, and Steven J. Schiff

***Abstract*—We describe a fast noniterative algorithm for the evaluation of continuous spline wavelet transforms at any integer scale $m$. In this approach, the input signal and the analyzing wavelet are both represented by polynomial splines. The algorithm uses a combination of moving sum and zero-padded filters, and its complexity per scale is $O(N)$, where $N$ is the signal length. The computation is exact, and the implementation is noniterative across scales. We also present examples of spline wavelets exhibiting properties that are desirable for either singularity detection (first and second derivative operators) or Gabor-like time-frequency signal analysis.**

## I. Introduction

The continuous wavelet transform (CWT) of a continuous-time signal $s(x)$ is defined by the convolution integral (cf. [1], [2])

$$W_{\psi}s(a,b) = \frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty} s(x)\psi\left(\frac{b-x}{a}\right)dx \tag{1}$$

where $\psi(x)$ is the analyzing wavelet, and $a$ and $b$ are continuously varying *shift* and *scale* parameters, respectively. Because of its unique localization and scaling properties, this representation provides an attractive tool for the analysis of signals with nonstationary characteristics [1], [3].

For implementation purposes, the parameters $a$ and $b$ are usually discretized. In the dyadic case, the analysis is restricted to scales that are powers of two: $a = 2^i$ and $i \in Z^+$. If, in addition, the wavelet $\psi$ is derived from a multiresolution analysis [4], it is possible to compute the wavelet transform in a very efficient fashion. In the critically sampled case, the discretization is $a = 2^i$, $b = 2^i k$, $k \in Z$, and one can use Mallat's algorithm, which relies on the use of two complementary low- and high-pass filters and has an overall $O(N)$ complexity [3], [4]. A similar strategy is also applicable for the oversampled case ($a = 2^i$, $b = k$) using zero-padded (or "à trous") filters [5], [6] with an $O(N)$ complexity per scale.

All the procedures described above are computationally very efficient and therefore widely used in practice. Unfortunately, they are limited to the computation of dyadic wavelet transforms. Here, we will consider an alternative scheme that allows for a finer discretization of the CWT at the integers ($a = m$, $b = k$) and has a complexity per scale that is comparable with that of Shensa's algorithm ($O(N)$). This greater freedom in the choice of $a$ is possible only because we are restricting ourselves to the class of polynomial spline CWT's and exploiting some very special scaling properties of the corresponding basis functions ($B$-splines). In addition, for a fixed integer value of $a$, the corresponding CWT is itself a polynomial spline and is therefore continuously known. Consequently, the method provides an exact computation. The disadvantages are minimal because splines are flexible enough to approximate virtually any desirable wavelet shape. In addition, splines provide the standard illustration of a multiresolution analysis; as a result, there are many examples of such wavelet bases in the literature [4], [7]. In addition, unlike other scaling functions and wavelets, splines have a simple explicit analytical form in both the time and frequency domain, which often facilitates their manipulations.

Manuscript received May 13, 1993; revised May 27, 1994. This work was supported, in part, by NIH grant IR29-MH 50006-01. The associate editor coordinating the review of this paper and approving it for publication was Prof. Ali N. Akansu.

M. Unser and A. Aldroubi are with the Biomedical Engineering and Instrumentation Program, National Center for Research Resources, National Institutes of Health, Bethesda, MD 20892 USA.

S. J. Schiff is with the Department of Neurosurgery, Children's National Medical Center, Washington, DC 20010 USA.

IEEE Log Number 9406015.

## II. Fast Wavelet Transform Algorithm

In our formulation, the input signal $s(x)$ and the wavelet $\psi(x)$ in (1) are both polynomial spline functions of degree $n_1$ and $n_2$, respectively. Let us therefore start by introducing the relevant spline representations and reviewing some of the important properties of $B$-spline basis functions.

### A. Basic Spline Properties

The splines considered here are constructed from polynomial segments of degree $n$ of unit length that are joined together at the knots[1] in a way that ensures the continuity of the resulting function and its derivatives up to order $(n-1)$. Such splines are characterized through their $B$-spline expansion

$$s(x) = \sum_{k\in Z} c(k)\beta^n(x-k) \tag{2}$$

where $\beta^n$ is the central $B$-spline of order $n$ and where the $c(k)$'s are the $B$-spline coefficients [8]. The $B$-splines are Gaussian-like functions of compact support that are generated from the repeated convolution of a $B$-spline of degree zero

$$\beta^n(x) = \beta^0 * \beta^{n-1}(x) \tag{3}$$

where $\beta^0(x)$ is the centered unit rectangular pulse.

The spline formalism provides a convenient mapping between the discrete and continuous signal domains [9]. Specifically, there is a one-to-one relation between a discrete signal $\{s[k]\}_{k\in Z}$ and its polynomial spline interpolant, which can be represented by the $B$-spline expansion (2). This mapping is expressed by the following discrete convolution equations [10]

$$s[k] := s(x)|_{x=k} = (b^n * c)(k) \Leftrightarrow c(k) = ((b^n)^{-1} * s)(k) \tag{4}$$

where

$$b^n(k) := \beta^n(x)|_{x=k} \tag{5}$$

is the discrete $B$-spline kernel of degree $n$ and where $(b^n)^{-1}$ denotes the inverse filtering operator.

Another key property is that there exists a simple mechanism for dilating the underlying basis functions. It is well known that a polynomial spline of degree $n$ (odd) enlarged by an integer factor $m$ is also a spline with (finer) knots at the integers. Therefore, there must exist a sequence $u_m^n(k)$ such that

$$\beta^n(x/m) = \sum_{k\in Z} u_m^n(k)\beta^n(x-k), \tag{6}$$

where the left-hand side represents a $B$-spline dilated by a factor of $m$. In fact, (6) is also valid for splines of even degree, provided that $m$ is odd.

[1] For $n$ odd, the knots are at the integers, whereas for $n$ even, they are at $k + 1/2$, $k \in Z$.

*Proposition 1:* The $z$-transform of the sequence $u_m^n(k)$, where $n$ and $m$ are not both even, is given by

$$U_m^n(z) = \frac{z^{k_0}}{m^n}\left(\sum_{k=0}^{m-1} z^{-k}\right)^{n+1} \tag{7}$$

with $k_0 = (n+1)(m-1)/2$.

Note that this result constitutes the continuous counterpart of the discrete $B$-spline convolution properties (3.5) and (3.6) described in [9].

*Proof:* A $B$-spline of order $n$ is obtained from the convolution of $(n+1)$ rectangular pulses. Therefore, the Fourier transform of a $B$-spline expanded by a factor of $m$ is

$$\beta^n(x/m) \overset{\text{Fourier}}{\longleftrightarrow} B_m^n(f) = m \cdot \text{sinc}^{n+1}(mf).$$

Next, we rewrite $B_m^n(f)$ as

$$B_m^n(f) = \frac{1}{m^n}\left(\frac{\sin \pi m f}{\sin \pi f}\right)^{n+1} \cdot \left(\frac{\sin \pi f}{\pi f}\right)^{n+1}$$

and note that the first of these factors has period one whenever the product $(n+1) \times (m-1)$ is even. The term $\sin(\pi m f)/\sin(\pi f)$ with an appropriate phase correction is then identified to be the Fourier transform of the discrete rectangular pulse with $z$-transform $\sum_{k=0}^{m-1} z^{-k}$. Therefore, we find that

$$B_m^n(f) = U_m^n(e^{j2\pi f}) \cdot \text{sinc}^{n+1}(f)$$

where $U_m^n(z)$ is given by (7) and where $k_0$ is an offset that ensures that $u_m^n$ is symmetrical. □

## B. A Digital Filtering Wavelet Algorithm

Let us now assume that the function to be analyzed $s(x)$ is a spline of degree $n_1$ specified by its $B$-spline representation (2) with $n = n_1$. Likewise, the analyzing wavelet $\psi(x)$ is a spline of degree $n_2$ with corresponding $B$-spline expansion

$$\psi(x) = \sum_{k \in Z} p(k)\beta^{n_2}(x-k). \tag{8}$$

Using (8) and Proposition 1, we show that the wavelet expanded by a factor of $m$ is given by

$$\psi_m(x) := \psi(x/m) = \sum_{k\in Z}([p]_{\uparrow m} * u_m^{n_2})(k)\beta^{n_2}(x-k) \tag{9}$$

where the notation $[p]_{\uparrow m}(k)$ represents the upsampling of the sequence $p$ by a factor of $m$ (i.e., insertion of $m-1$ zeros in between taps). Next, we recall that the convolution of two splines of degree $n_1$ and $n_2$ is a spline of degree $(n_1+n_2+1)$ and that this operation can be implemented by a simple convolution in the discrete $B$-spline domain (cf. Section IV-B of [9]). Therefore, the continuous wavelet transform of $s(x)$ at scale $m$ is given by

$$W_\psi s(m,b) = (\psi_m * s)(b) = \sum_{k\in Z}([p]_{\uparrow m} * u_m^{n_2} * c)(k) \times \beta^{n_1+n_2+1}(b-k) \tag{10}$$

which is the $B$-spline representation of a spline of degree $(n_1 + n_2+1)$. This formula provides an exact representation of the wavelet transform at scale $m$ as a function of the continuous shift parameter $b$.

For visualization purposes, a representation in terms of sampled values is usually more appropriate. To obtain these samples, it is sufficient to consider the values of the $B$-spline basis functions in (10) at the integers. The whole computation is therefore equivalent to the following cascade of convolutions:

$$w_m(k) := W_\psi s[m,k] = ([p]_{\uparrow m} * u_m^{n_2} * b^{n_1+n_2+1} * c)(k) \tag{11}$$

where $b^{n_1+n_2+1}$ is the discrete $B$-spline of order $(n_1+n_2+1)$ (cf. (5)). The samples $w_m(k)$ also provide a unique characterization of the wavelet transform (10) at scale $m$. In fact, it is easy to recover the $B$-spline coefficients using the inverse mapping described by (4).

## C. Fast Implementation

The fast implementation of (10) for any integer scale $m$ is achieved in three steps, which are described next. The whole procedure is schematically represented in Fig. 1.

i) *Initialization*: This part of the calculation is only performed once. It combines the evaluation of the $B$-spline coefficients $c(k)$ that interpolate the input signal $s(k)$ (cf. (4)) and the convolution with the kernel representing the basis functions for the wavelet transform (cf. (11)). We thereby compute the auxiliary signal

$$\begin{aligned} s_1(k) :&= \langle s(x), \beta^{n_2}(x-k)\rangle \\ &= b^{n_1+n_2+1} * (b^{n_1})^{-1} * s[k] \cong b^{n_2} * s[k]. \end{aligned} \tag{12}$$

The filter $b^{n_1+n_2+1}$ is a symmetric finite impulse response (FIR) kernel, which is characterized by a vector $\boldsymbol{b}$ of size $n_b$ that contains the filter coefficients (cf. Table I). The inverse filter $(b^{n_1})^{-1}$ can be implemented very efficiently using the recursive algorithm described in [10]. Note that this operation is an identity for $n_1 = 1$ (piecewise linear signal model). Alternatively, we may use the right-hand side formula, which provides the discrete approximation of the $L_2$-inner product.

ii) *Iterated Moving Sum*: The next step is to compute the quantities

$$s_m(k) := \langle s(x), \beta^{n_2}((x-k)/m)\rangle = u_m^{n_2} * s_1(k). \tag{13}$$

This filtering can be done in a very efficient manner since it is equivalent to a cascade of $(n_2+1)$ moving sums (cf. (7)). Specifically, we have that

$$u_m^{n_2}(k) = \frac{1}{m^{n_2}} \underbrace{u_m^0 * u_m^0 \cdots * u_m^0}_{(n_2+1)\text{ times}}(k) \tag{14}$$

where the moving sum filter is defined as follows:

$$r_i(k) = u_m^0 * r_{i-1}(k) = \sum_{l=k+l_0+1}^{k+l_0+m} r_{i-1}(l) \tag{15}$$

and where $l_0$ is an appropriate offset. Clearly, this filter can be evaluated with two additions per sample using a recursive updating strategy

$$r_i(k) = r_i(k-1) + r_{i-1}(k+l_0+m) - r_{i-1}(k+l_0) \tag{16}$$

with the initial condition $r_0(k) = s_1(k)$. Therefore, the computational cost for evaluating (13) is only of $2(n_2+1)$ additions per sample, irrespective of the value of the parameter $m$. Note that all multiplicative factors to be applied to the wavelet coefficients (the factors $1/m^{n_2}$ in (14) and $1/\sqrt{m}$ in (1)) can be included in the last filtering step by appropriate rescaling of the filtering template $p$.

iii) *Zero-Padded Filter*: Last, the wavelet coefficients are obtained by filtering with the up-sampled kernel representing the $B$-spline coefficients of the wavelet $\psi$, i.e.

$$w_m(k) = [p]_{\uparrow m} * s_m(k). \tag{17}$$

The FIR operator $p$ is characterized by a vector $\boldsymbol{p}$ of size $n_p$ (cf. Table I). Thus, the complexity of this part of the algorithm is $n_p/2$ multiplications and $n_p - 1$ additions per sample, assuming that $p$ is either symmetric or antisymmetric.

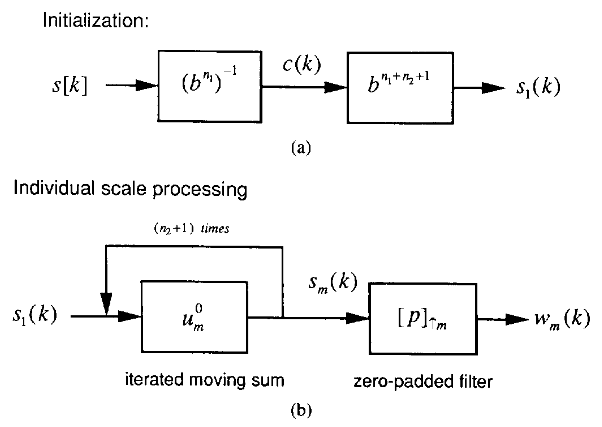

Fig. 1. Schematic representation of the fast wavelet transform algorithm: (a) Initialization; (b) individual scale processing.

TABLE I
FILTER PARAMETERS FOR THE FAST WAVELET TRANSFORM ALGORITHM

| Filter/wavelet type | Template |
|---|---|
| *Discrete B-splines*: | |
| Cubic spline ($n$=3) | $b^3=\frac{1}{6}(\bullet, 4, 1)$ |
| Quintic spline ($n$=5) | $b^5=\frac{1}{120}(\bullet, 66, 26, 1)$ |
| 7th order spline ($n$=7) | $b^7=\frac{1}{5040}(\bullet, 2416, 1191, 120, 1)$ |
| *Cubic spline wavelets* ($n_2$=3): | |
| 1st derivative | $p_I=(-1,+1)$, $p_I^{(2)}=(-1,-4,-5, 0,+5,+4,+1)$ |
| 2nd derivative | $p_{II}=(\bullet, 2,-1)$ |
| B-spline wavelet | $p_{III}=(\bullet, 0.6018, -0.4584, 0.196, -0.04159, 0.003075,-0.0000248)$ |
| *Examples of non-wavelet filters* ($n_2$=3): | |
| Quasi-gaussian | $p_G=(1)$ (identity) |
| Lowpass (cardinal spline) | $p^3_{card}=(\bullet, 1.7321, -0.4641, 0.12436, -0.03332, 0.00893, -0.00239,0.0006410, -0.0001718, \ldots)$ |

Note : The symbol • indicates that the sequence is symmetrical.

## III. RESULTS AND DISCUSSION

The present algorithm allows for an exact computation of polynomial spline wavelet transforms with integer scales. If the desired wavelet prototype is not a spline, we can still approximate it by constructing its least squares approximation [11] or by simply interpolating the function values at the integers. In practice, however, the exact shape of the wavelet is not necessarily the most important factor. It is usually preferable to identify a specific property that is particularly relevant for the application at hand. For instance, if the goal is to analyze and characterize singularities, then the wavelet should correspond to the first (or second) derivative of a smoothing function that is well localized in time and frequency [12]. A typical example is the Canny operator, which detects edges from the maxima of the gradient [13]. Alternatively, singularities may also be found from the zero-crossings of a smoothed Laplacian (second derivative operator) such as the Marr–Hildreth or mexican hat detector [14]. If, on the other hand, the goal is to perform some kind of time-frequency signal analysis, then the choice of a wavelet with (near) optimal localization is more appropriate. Several such examples of spline wavelets are presented next.

### A. Examples of Spline Wavelets

• *First and Second Derivative Wavelets*: $B$-splines of higher degrees are particularly well localized in both time and frequency because they converge to a Gaussian as $n$ goes to infinity (central limit theorem). For $n$ =3, the time-frequency bandwidth product is already within 0.5% of the limit specified by the uncertainty principle [15]. It is therefore justified to select $B$-spline smoothing functions for the design of derivative operators.

The first-order derivative of a $B$-spline (with a change of sign) is given by (cf. [9])

$$\psi_I(x) = -\frac{d\beta^n(x)}{dx} = -\beta^{n-1}\left(x+\frac{1}{2}\right) + \beta^{n-1}\left(x-\frac{1}{2}\right). \quad (18)$$

It corresponds to a particular instance of (8) with $\boldsymbol{p}_{\rm I} = (-1, 1)$, provided that we shift the origin by half a sampling step. Alternatively, if we want an operator that is truly antisymmetric, we can take the derivative of a $B$-spline enlarged by a factor of two, which leads to the sequence $\boldsymbol{p}_{\rm I}^{(2)} = (-1, -4, -5, 0, 5, 4, 1)$. The graph of this wavelet for $n_2 = 3$ is given in Fig. 2(a). Also represented is the derivative of a Gaussian that is frequently used as an approximation to the optimal Canny detector [13]. These functions are all qualitatively very similar.

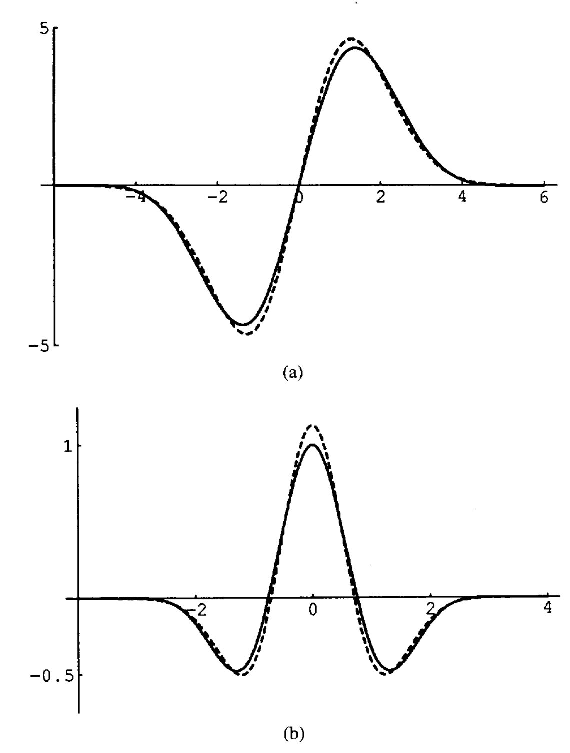

Fig. 2. Examples of cubic spline derivative wavelets (solid line): (a) First derivative of a fourth-order $B$-spline expanded by a factor of two and its Gaussian (first derivative) approximation (dashed line); (b) second derivative of a quintic $B$-spline and its mexican hat approximation (dashed line).

The second derivative of a $B$-spline of order $n$ (with a change a sign) is a symmetrical spline of order $n-2$ that is given by (cf. [9])

$$\begin{aligned}\psi_{\rm II}(x) &= -\frac{d^2\beta^n(x)}{dx^2} \\ &= -\beta^{n-2}(x+1) + 2\beta^{n-2}(x) - \beta^{n-2}(x-1). \quad (19)\end{aligned}$$

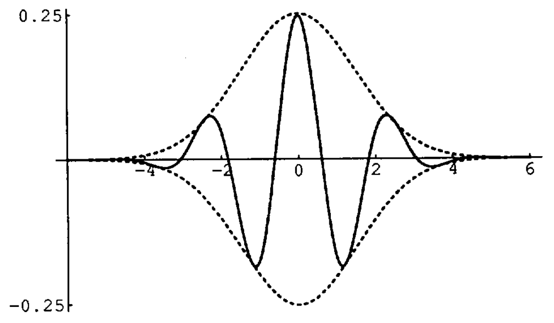

Fig. 3. Cubic $B$-spline wavelet (solid line) and its Gaussian envelope (dashed line).

The corresponding $B$-spline coefficients are $\boldsymbol{p}_{\text{II}} = (-1, 2, -1)$. The second derivative cubic spline wavelet is shown in Fig. 2(b) together with its mexican hat approximation (second derivative of a Gaussian):

• *$B$-Spline Wavelets* [15], [16]: These functions are perfect candidates for time-frequency signal analyses because of their excellent localization properties; they have been shown to converge to cosine-Gabor functions as the order of the spline increases [15]. The $B$-spline wavelets are also compactly supported and are splines by construction. The corresponding weighting sequence $\boldsymbol{p}_{\text{III}}$ for $n_2$=3 is given in Table I. The cubic $B$-spline wavelet and the Gaussian envelope of its Gabor approximation given by (2.9) in [15] are shown in Fig. 3. This wavelet is unique in the sense that it can be used to construct dyadic wavelet bases of $L_2$; it also has a simple reconstruction algorithm associated with it [17].

## B. Additional Remarks

The overall complexity of the algorithm for these various spline wavelets is $O(N \cdot M)$, where $N$ is the total number of samples, and $M$ is the number of desired scales, which is a factor $\log N$ improvement over comparable FFT-based approaches [18]. Overall, the algorithm is relatively straightforward to implement. To avoid discontinuities at the boundaries, we have extended the signal on both ends by using mirror reflections: a standard practice in signal processing. In addition, we have chosen $n_2$ to be odd in order to be able to apply the algorithm for all positive integer scales $m$; otherwise, the procedure would be valid for odd dilation factors only (c.f. Proposition 1).

There are a few practical issues in the computation of (13) using iterated moving sums. First, the sums have to be initialized at the starting position at the cost of $m$ additional operations. Second, the offset parameter $l_0$ in (15) must be chosen in an appropriate way. When $m$ is odd, all windows are simply centered on the current position. When $m$ is even, the window can no longer be centered, and we use a simple alternating shifting scheme to produce a response that is globally symmetrical; this is possible because the number of iterations is even.

We have implemented the algorithm using the Pro-Matlab software with some of the critical subroutines (iterated sum and zero-padded filter) coded in $C$ to speed up computations. The area of application in which we have been particularly involved is the analysis of EEG recordings for seizure detection [19]. Thanks to the new algorithm, we were able to reduce the CPU time for processing a 6144-point EEG signal with 64 scales from several days on a Sun SparcStation IPx (using a direct $O(N^2)$ method, as described in [20]) to 30 s on a Macintosh IIfx computer. Finally, we note that the present approach may also provide very efficient computational solutions for other signal processing tasks such as scale-space signal analysis and narrow-band low-pass filtering (cf. the two last filter examples in Table I). The last set of coefficients corresponds to the cardinal (or fundamental) spline; a function that converges to the ideal low-pass filter as $n$ goes to infinity [21].

## IV. Conclusion

The spline CWT algorithm that has been described has the following attractive features:

- It can handle any integer dilation factor and is not restricted to powers of two.
- The complexity per scale is $O(N)$ with a proportionality constant that can be surprisingly small. For example, the second derivative cubic spline CWT requires no more than 10 additions plus two multiplications per output sample (neglecting the initialization phase which is only performed once).
- For spline wavelets, the computation is exact, and the CWT at scale $m$ is continuously known.
- The algorithm is noniterative across scales, and its structure is very regular. It is therefore well suited for a parallel implementation with one processor per scale.
- The procedure offers flexibility in the choice of wavelet shapes, as illustrated by our examples.

## References

[1] I. Daubechies, *Ten Lectures on Wavelets.* Philadelphia: Soc. Indus. Applied Math., 1992.

[2] A. Grossmann and J. Morlet, "Decomposition of Hardy functions into square integrable wavelets of constant shape," *SIAM J. Math. Anal.*, vol. 15, no. 4, pp. 723–736, July 1984.

[3] O. Rioul and M. Vetterli, "Wavelets and signal processing," *IEEE Signal Processing Mag.*, vol. 8, no. 4, pp. 11–38, Oct. 1991.

[4] S. G. Mallat, "A theory of multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 11, no. 7, pp. 674–693, 1989.

[5] M. J. Shensa, "The discrete wavelet transform: wedding the à trous and Mallat algorithms," *IEEE Trans. Signal Processing*, vol. 40, no. 10, pp. 2464–2482, Oct. 1992.

[6] O. Rioul and P. Duhamel, "Fast algorithms for discrete and continuous wavelet transforms," *IEEE Trans Inform. Theory*, vol. 38, no. 2, pp. 569–586, Mar. 1992.

[7] M. Unser and A. Aldroubi, "Polynomial splines and wavelets—A signal processing perspective," in *Wavelets—A Tutorial in Theory and Applications* (C. K. Chui, Ed.). San Diego: Academic, 1992, pp. 91–122.

[8] I. J. Schoenberg, "Contribution to the problem of approximation of equidistant data by analytic functions," *Quart. Appl. Math.*, vol. 4, pp. 45–99, 112–141, 1946.

[9] M. Unser, A. Aldroubi, and M. Eden, " $B$-spline signal processing. Part I: Theory," *IEEE Trans. Signal Processing*, vol. 41, no. 2, pp. 821–833, Feb. 1993.

[10] ——, "Fast $B$-spline transforms for continuous image representation and interpolation," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 13, no. 3, pp. 277–285, Mar. 1991.

[11] ——, "Polynomial spline signal approximations: Filter design and asymptotic equivalence with Shannon's sampling theorem," *IEEE Trans. Inform. Theory*, vol. 38, no. 1, pp. 95–103, Jan. 1992.

[12] S. Mallat and W. L. Hwang, "Singularity detection and processing with wavelets," *IEEE Trans. Inform. Theory*, vol. 38, no. 2, pp. 617–643, Mar. 1992.

[13] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-8, no. 6, pp. 679–697, 1986.

[14] D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Roy. Soc. London*, vol. B 207, pp. 187–217, 1980.

[15] M. Unser, A. Aldroubi, and M. Eden, "On the asymptotic convergence of $B$-spline wavelets to Gabor functions," *IEEE Trans. Inform. Theory*, vol. 38, no. 2, pp. 864–872, Mar. 1992.

[16] C. K. Chui and J. Z. Wang, "On compactly supported spline wavelets and a duality principle," *Trans. Amer. Math. Soc.*, vol. 330, no. 2, pp. 903–915, 1992.

[17] M. Unser, A. Aldroubi, and M. Eden, "A family of polynomial spline wavelet transforms," *Signal Processing*, vol. 30, no. 2, pp. 141–162, Jan. 1993.

[18] D. L. Jones and R. G. Baraniuk, "Efficient approximation of continuous wavelet transforms," *Electron. Lett*, vol. 27, no. 9, pp. 748–750, Apr. 1991.

[19] S. Schiff, A. Aldroubi, M. Unser, and S. Sato, "Fast wavelet transform of EEG," *Electroencephalog. Clin. Neurophysiol.*, in press.

[20] S. J. Schiff, J. Heller, S. L. Weinstein, and J. Milton, "Wavelet transforms and surrogate data for electroencephalographic spike and seizure detection," *Opt. Eng.*, vol. 33, pp. 2162–2169, 1994.

[21] A. Aldroubi, M. Unser, and M. Eden, "Cardinal spline filters: Stability and convergence to the ideal sinc interpolator," *Signal Processing*, vol. 28, no. 2, pp. 127–138, Aug. 1992.

# A Note on the Error Propagation Analysis of Recursive Least Squares Algorithms

Bin Yang

*Abstract*— **This correspondence comments on the error propagation analysis of recursive least squares algorithms presented by Ljung and Ljung, Verhaegen, and Haykin. In particular, it corrects some incorrect statements about the error propagation given in the literature.**

## I. Introduction

Recursive least squares (RLS) algorithms are used in a broad class of applications. Since some internal variables of these algorithms are computed recursively, round-off errors of past and present computations propagate into future time instants resulting in error accumulation. Due to the potential danger of divergence, there has been much interest to study the error propagation and accumulation of RLS algorithms.

Ljung and Ljung [1] have presented possibly the first rigorous single-error propagation analysis. However, they studied only the error propagation in the recursive equation for updating the weight vector. Verhaegen [2] has given a more detailed single-error propagation analysis. He also investigated the error propagation through the inverse correlation matrix. His work reveals the root of explosive divergence of the classical RLS algorithm and proposes a stabilization scheme. Haykin summarized in [3] the main results of both works.

We find, however, that some of their statements are not precise or wrong. We will show that the propagation of a single error through the weight vector or the inverse correlation matrix is not contractive for the growing memory RLS algorithm ($\lambda = 1$) even if the input data is persistently exciting. The error may grow in norm, but the growth is bounded. This noncontractive error propagation explains the divergence of the growing memory RLS algorithm in finite-precision computations [4].

In the following, an underlined character represents a column vector, and a boldface character signifies a matrix. The superscript $^T$ describes transposition. $\|\ \|$ denotes the 2-norm for both vectors and matrices.

Manuscript received September 19, 1993; revised July 4, 1994. The associate editor coordinating the review of this paper and approving it for publication was Dr. Fuyun Ling.

The author is with the Department of Electrical Engineering, Ruhr University Bochum, Bochum, Germany.

IEEE Log Number 9406020.

TABLE I
Summary of Version I of the RLS Algorithm

| | | | |
|---|---|---|---|
| $\mathbf{P}(0) = \delta^{-1}\mathbf{I},\ \underline{w}(0) = \underline{0}$ | | | |
| FOR $t = 1, 2, \ldots$ DO | $\pm$ | $\times$ | $\div$ |
| 1) $\underline{h}(t) = \mathbf{P}(t-1)\underline{x}(t)$ | $N(N-1)$ | $N^2$ | |
| 2) $\alpha(t) = 1/[\lambda + \underline{x}^T(t)\underline{h}(t)]$ | $N$ | $N$ | 1 |
| 3) $\underline{g}(t) = \underline{h}(t)\alpha(t)$ | | $N$ | |
| 4) $\mathbf{P}(t) = \lambda^{-1}\left[\mathbf{P}(t-1) - \underline{g}(t)\underline{h}^T(t)\right]$ | $N^2$ | $2N^2$ | |
| 5) $e(t, t-1) = y(t) - \underline{w}^T(t-1)\underline{x}(t)$ | $N$ | $N$ | |
| 6) $\underline{w}(t) = \underline{w}(t-1) + \underline{g}(t)e(t, t-1)$ | $N$ | $N$ | |
| END $\qquad\Sigma$ | $2N^2 + 2N$ | $3N^2 + 4N$ | 1 |

## II. RLS Algorithms

We consider the problem of spatial adaptive filtering using the RLS minimization. Let $x_i(t)$ $(i = 1, 2, \ldots, N)$ be the observations of $N$ input signals at the discrete time instant $t \geq 1$. Using a linear combination of them, a desired signal $y(t)$ has to be estimated. In vector notations, the estimate is $\hat{y}(t) = \underline{w}^T(t)\underline{x}(t)$, where $\underline{x}(t) = [x_1(t), x_2(t), \ldots, x_N(t)]^T$ is the $N \times 1$ input data vector, and $\underline{w}(t) = [w_1(t), w_2(t), \ldots, w_N(t)]^T$ is the weight vector involved.

The purpose of least squares estimation is to choose $\underline{w}(t)$ to minimize the sum of exponentially weighted squared errors $\mathcal{E}(t) = \sum_{i=1}^{t} \lambda^{t-i}[y(i) - \underline{w}^T(t)\underline{x}(i)]^2$. $0 < \lambda \leq 1$ is the forgetting factor. It is intended to ensure that data in the distant past are "forgotten" in order to provide the tracking capability when the system operates in a nonstationary environment. The solution of the above problem is well known. Let

$$\mathbf{C}(t) = \sum_{i=1}^{t} \lambda^{t-i}\underline{x}(i)\underline{x}^T(i) = \lambda\mathbf{C}(t-1) + \underline{x}(t)\underline{x}^T(t) \qquad (1)$$

be the correlation matrix of the input data vector, and let $\underline{\Delta}(t) = \sum_{i=1}^{t} \lambda^{t-i}\underline{x}(i)y(i)$ be the cross-correlation vector between $\underline{x}(t)$ and $y(t)$, respectively. The weight vector minimizing $\mathcal{E}(t)$ is given by $\underline{w}(t) = \mathbf{C}^{-1}(t)\underline{\Delta}(t)$, provided that $\mathbf{C}(t)$ is of full rank.

There exist various algorithms to compute $\underline{w}(t)$ recursively. We consider, in this correspondence, RLS algorithms based on the application of the matrix inversion lemma to (1). In the literature, different versions of the RLS algorithm are available [1]–[3], [5]. Tables I to III summarize three widely known versions for convenience of discussions. The first two RLS schemes in Tables I and II are discussed in detail in [2], [3]. They are referred to as versions I and II. Version III of the RLS algorithm in Table III is the most efficient one because it computes only the upper (or lower) triangular part of $\mathbf{P}(t)$ as indicated by the operator Tri{}. Note that $\mathbf{P}(t) = \mathbf{C}^{-1}(t)$ is the inverse correlation matrix. $\underline{g}(t) = \mathbf{P}(t)\underline{x}(t)$ is the so-called gain vector, and $e(t, t-1)$ is the *a priori* estimation error. The total operation counts listed in the tables are given for the case $\lambda < 1$ and that all divisions by $\lambda$ are replaced by multiplications with the precalculated factor $\lambda^{-1}$.

## III. Error Propagation Analysis

### A. Error Propagation Through the Weight Vector

The propagation of a single error in the weight vector to subsequent recursions in infinite-precision computations has been studied in [1]–[3]. We first recall some of the results of these studies.

Let $\overline{\underline{w}}(t_0) = \underline{w}(t_0) + \delta\underline{w}(t_0)$ be the erroneous version of $\underline{w}(t_0)$. The error vector $\delta\underline{w}(t_0)$ describes a perturbation to $\underline{w}(t_0)$, which may arise from the quantization of $\underline{w}(t_0)$. If all other variables of