BASEBAND VOLTERRA FILTERS FOR IMPLEMENTING CARRIER BASED NONLINEARITIES

Gil M. Raz and Barry D. Van Veen, Member, IEEE

Department of Electrical and Computer Engineering
University of Wisconsin-Madison
1415 Engineering Drive, Madison, WI 53706 USA
email: raz@ece.wisc.edu vanveen@engr.wisc.edu

ABSTRACT

A diagonal coordinate representation for Volterra filters is developed and exploited to derive efficient Volterra filter implementations for processing carrier based input signals. In the diagonal coordinate representation the output is expressed as a sum of linear filters applied to modified input signals. Hence, linear filtering methods are employed to implement the nonlinear filter on a baseband version of the input. Downsampling is then used to reduce computational complexity.

1. INTRODUCTION

The Volterra filter [5] is one of the most widely used nonlinear system representations, in large part because the output is a linear function of the filter parameters. A causal, stable, time-invariant, finite memory, discrete-time system may be represented in terms of the Volterra filter output

$$y(k) = \sum_{n=1}^{N} y_n(k) \tag{1}$$

where

$$y_n(k) = \sum_{k_1=0}^{m-1} \cdots \sum_{k_n=0}^{m-1} h_n(k_1, \ldots, k_n) \prod_{i=1}^{n} u(k-k_i) \tag{2}$$

Here m is the memory length, $h_n(k_1, \ldots, k_n)$ is the n^{th} order kernel, and $u(k)$ is the input. The kernels can be assumed symmetric with respect to any permutation of the independent variables without loss of generality.

One of the problems inherent to the Volterra representation is the computational complexity involved in calculating the output due to the large number of parameters in the Volterra kernels. There are several methods of reducing computational complexity using approximated Volterra filters.

This paper develops a computationally efficient method for exact Volterra filter implementation by assuming the input to the system is band limited. Band limited inputs frequently occur in communication systems applications of Volterra filters, such as arise in equalization of nonlinear channels [4, 1]. The computationally efficient implementation presented here is obtained by expressing the Volterra filter in terms of a diagonal coordinate system. The output is then given by a sum of linear filter outputs operating on nonlinear combinations of the input. Down-sampling is used to decrease the computational cost of implementing the linear filters. The diagonal coordinate representation also offers clear insight into the relationship between the characteristics of the output in the frequency domain and the filter parameters. This interpretation offers significant advantages over the multidimensional frequency domain interpretations proposed in [2] for many problems.

The outline of this work is as follows. In Section II the diagonal coordinate representation for the Volterra filter is derived. The down-sampling based computationally efficient implementation for band limited input is derived and analyzed in Section III. The paper concludes with a summary.

2. DIAGONAL COORDINATE REPRESENTATION FOR DISCRETE TIME FINITE MEMORY VOLTERRA FILTERS

It is insightful to rewrite the Volterra filter output in terms of the diagonal elements of the kernel. Begin by removing the redundant summation indices in (2) associated with the kernel symmetry. We write

$$\begin{aligned} y_n(k) &= \sum_{k_1=0}^{m-1} \sum_{k_2=k_1}^{m-1} \cdots \sum_{k_n=k_{n-1}}^{m-1} h_n(k_1, \ldots, k_n) \\ &\quad \times C(k_1, \ldots, k_n) \prod_{i=1}^{n} u(k-k_i) \end{aligned} \tag{3}$$

where $C(k_1, \ldots, k_n)$ is the number of different possible permutations of the set of numbers $k_1, \ldots, k_n$.

Now introduce the change of coordinates

$$\begin{aligned} k_1 &= s \\ k_i &= s + r_{i-1} \quad \forall i = 2, \ldots, n \end{aligned} \tag{4}$$

so that the output of the n^{th} order kernel is written as

$$y_n(k) = \sum_{r_1=0}^{m-1} \sum_{r_2=r_1}^{m-1} \cdots \sum_{r_{n-1}=r_{n-2}}^{m-1} \sum_{s=0}^{m-1-r_{n-1}} h_n(s, s+r_1, \ldots, s+r_{n-1})$$

Supported by the Army Research Office under Grant DAAH04-93-G-0208.

$$\times C(0, r_1, \ldots, r_{n-1}) u(k-s) \prod_{i=1}^{n-1} u(k-s-r_i) \quad (5)$$

Define the new signals

$$v_{r_1,\ldots,r_{n-1}}(k) = u(k) \prod_{i=1}^{n-1} u(k-r_i) \quad (6)$$

and filters

$$g_{r_1,\ldots,r_{n-1}}(k) = C(0, r_1, \ldots, r_{n-1}) h_n(k, k+r_1, \ldots, k+r_{n-1}) \quad (7)$$

Hence, (5) may be rewritten as

$$y_n(k) = \sum_{r_1=0}^{m-1} \sum_{r_2=r_1}^{m-1} \cdots \sum_{r_{n-1}=r_{n-2}}^{m-1} v_{r_1,\ldots,r_{n-1}}(k) * g_{r_1,\ldots,r_{n-1}}(k) \quad (8)$$

where $*$ is the convolution operator.

Here we have expressed the output of the n^{th} order kernel as a sum of one-dimensional convolutions. A one-dimensional frequency domain description for the n^{th} order kernel output is obtained by taking the discrete-time Fourier transform of (8)

$$Y_n(\omega) = \sum_{r_1=0}^{m-1} \sum_{r_2=r_1}^{m-1} \cdots \sum_{r_{n-1}=r_{n-2}}^{m-1} V_{r_1,\ldots,r_{n-1}}(\omega) G_{r_1,\ldots,r_{n-1}}(\omega) \quad (9)$$

Here $V_{r_1,\ldots,r_{n-1}}(\omega)$ and $G_{r_1,\ldots,r_{n-1}}(\omega)$ are the discrete-time Fourier transforms of $v_{r_1,\ldots,r_{n-1}}(k)$ and $g_{r_1,\ldots,r_{n-1}}(k)$ respectively.

While it is useful to think in terms of diagonal coordinates, it forces a rather cumbersome notation. Let $D(n,m)$ be the number of non-redundant diagonals, paralleling the main diagonal, in the kernel $h_n(k_1, \ldots, k_n)$. We may express $D(n,m)$ in closed form [3] as

$$D(n,m) = \frac{(n+m-2)!}{(n-1)!(m-1)!} \quad (10)$$

Now define $\mathbf{r}_j$, $j = 1, 2, \ldots, D(n,m)$ as the $(n-1)$-tuple $[r_1, \ldots, r_{n-1}]$ corresponding to the j^{th} diagonal of $h_n(k_1, \ldots, k_n)$ so that we may rewrite (8) and (9) as

$$y_n(k) = \sum_{j=1}^{D(n,m)} g_{\mathbf{r}_j}(k) * v_{\mathbf{r}_j}(k) \quad (11)$$

and

$$Y_n(\omega) = \sum_{j=1}^{D(n,m)} G_{\mathbf{r}_j}(\omega) V_{\mathbf{r}_j}(\omega) \quad (12)$$

respectively.

We refer to (11) as a serial implementation of a homogeneous Volterra filter of order n. The computational complexity of this implementation is determined as follows. The average length of the impulse responses $g_{\mathbf{r}_j}(k)$ is

$$N_{\text{length}} = \left(\sum_{r_1=0}^{m-1} \sum_{r_2=r_1}^{m-1} \cdots \sum_{r_{n-1}=r_{n-2}}^{m-1} (m - r_{n-1}) \right) / (D(n,m)) = \frac{n+m-1}{n} \quad (13)$$

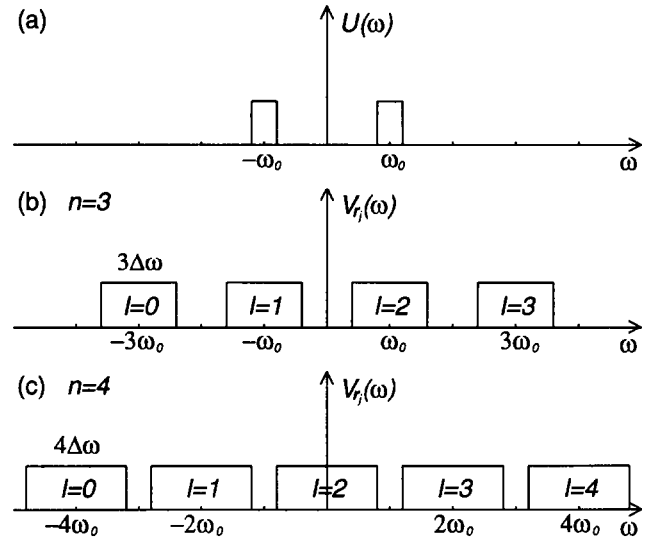

Figure 1: Frequency support of input and modified input.

The number of real multiplications required to calculate $v_{\mathbf{r}_j}(k)$, $j = 1, 2, \ldots, D(n,m)$ is approximately bounded by $2D(n,m)$ [3]. Hence, the overall number of real multiplications required to compute each output value is $M_{\text{serial}} = D(n,m)(2 + N_{\text{length}})$, assuming the convolution is implemented in the time domain.

The diagonal coordinate representation is particularly useful interpreting systems that are output band limited because the output frequency content is directly related to the frequency response of the diagonal elements of the kernel. In the following section we extend this interpretation to include knowledge of the input frequency support.

3. EFFICIENT IMPLEMENTATION FOR CARRIER BASED INPUT

Let the input to the nonlinear system, $u(k)$ be a band limited with bandwidth $\Delta\omega$ and center frequency ω_0, as depicted in Figure 1 (a). We assume all frequencies are normalized to the interval $[-\pi, \pi]$, with $\omega = \pi$ representing the Nyquist frequency. The output of an n^{th} order nonlinear system may have energy at frequencies up to n times the highest input frequency. Hence, to avoid aliasing in the system output we require $n(\omega_0 + \frac{\Delta\omega}{2}) \leq \pi$. For sake of simplicity, we shall assume $n(\omega_0 + \frac{\Delta\omega}{2}) = \pi$.

While carrier based signals are often continuous functions of time, the discrete-time approach followed in this section is instructive and leads to an efficient discrete-time implementation for sampled continuous-time signals.

3.1. Frequency Domain Interpretation

Equation (12) indicates that the band of frequencies in which the output $y_n(k)$ lies is limited to the bands for which $v_{\mathbf{r}_j}(k)$, $j = 1, 2, \ldots, D(n,m)$ contains energy. Hence, we relate the frequency support of the input $u(k)$ to that of the $v_{\mathbf{r}_j}(k)$. The DTFT of the input, $U(\omega)$, has two

bands of energy in the range $[-\pi, \pi]$. They are, $I^- = [-\omega_0 - \frac{\Delta\omega}{2}, -\omega_0 + \frac{\Delta\omega}{2}]$ and $I^+ = [\omega_0 - \frac{\Delta\omega}{2}, \omega_0 + \frac{\Delta\omega}{2}]$. We define

$$U^+(\omega) = \begin{cases} U(\omega) & \text{for } \omega \in I^+ \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

$$U^-(\omega) = \begin{cases} U(\omega) & \text{for } \omega \in I^- \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

Clearly, $U(\omega) = U^+(\omega) + U^-(\omega)$.

For a given $i \in \{1, \ldots, D(n,m)\}$ we write $\mathbf{r}_i = [r_1, \ldots, r_{n-1}]$, and thus do not explicitly indicate the dependence of the elements $r_1, \ldots, r_n$ on i to simplify the notation. The DTFT of $u(k-r_\alpha)$ is $U(\omega)\exp(-j\omega r_\alpha)$, which is written as $U^+_{r_\alpha}(\omega) + U^-_{r_\alpha}(\omega)$ by defining $U^+_{r_\alpha}(\omega) = U^+(\omega)\exp(-j\omega r_\alpha)$ and $U^-_{r_\alpha}(\omega) = U^-(\omega)\exp(-j\omega r_\alpha)$. Thus, we have

$$\begin{aligned} V_{\mathbf{r}_i}(\omega) &= U(\omega) * U(\omega)\exp(-j\omega r_1) * \cdots \\ &\quad * U(\omega)\exp(-j\omega r_{n-1}) \\ &= (U^+(\omega) + U^-(\omega)) * (U^+_{r_1}(\omega) + U^-_{r_1}(\omega)) * \cdots \\ &\quad * (U^+_{r_{n-1}}(\omega) + U^-_{r_{n-1}}(\omega)) \end{aligned} \tag{16}$$

Distribute the convolution over addition to write

$$V_{\mathbf{r}_i}(\omega) = \sum_{l=0}^{n} V_{\mathbf{r}_i,l}(\omega) \tag{17}$$

where $V_{\mathbf{r}_i,l}(\omega)$ involves convolutions of l terms $U^+_{r_j}(\omega)$ and $n-l$ terms $U^-_{r_j}(\omega)$. That is,

$$\begin{aligned} V_{\mathbf{r}_i,0}(\omega) &= U^-(\omega) * U^-_{r_1}(\omega) * \cdots * U^-_{r_{n-1}}(\omega) \\ V_{\mathbf{r}_i,1}(\omega) &= U^+(\omega) * U^-_{r_1}(\omega) * \cdots * U^-_{r_{n-1}}(\omega) + \\ &\quad U^-(\omega) * U^+_{r_1}(\omega) * U^-_{r_2}(\omega) * \cdots * U^-_{r_{n-1}}(\omega) + \\ &\quad \vdots \\ &\quad U^-(\omega) * U^-_{r_1}(\omega) * \cdots * U^-_{r_{n-2}}(\omega) * U^+_{r_{n-1}}(\omega) \\ &\vdots \\ V_{\mathbf{r}_i,n}(\omega) &= U^+(\omega) * U^+_{r_1}(\omega) * \cdots * U^+_{r_{n-1}}(\omega) \end{aligned} \tag{18}$$

By grouping the terms in this manner, we may identify the frequency band containing energy for each $V_{\mathbf{r}_i,l}(\omega)$. Since all terms in $V_{\mathbf{r}_j,l}(\omega)$ are a convolution of l terms from I^+ and $n-l$ terms from I^-, $V_{\mathbf{r}_i,l}(\omega)$ has center frequency $l\omega_0 + (n-l)(-\omega_0) = (2l-n)\omega_0$ and its energy is limited to the frequency band

$$I_l = \left(\omega_0(2l-n) - n\frac{\Delta\omega}{2}, \omega_0(2l-n) + n\frac{\Delta\omega}{2}\right) \tag{19}$$

as illustrated in Figure 1 (b) and (c) for $n=3$ and $n=4$, respectively. The frequency bands I_l overlap only if $n\Delta\omega \geq 2\omega_0$.

While the filter $G_{\mathbf{r}_i}(\omega)$ may have nonzero response over the entire interval $[-\pi, \pi]$, we need only consider its behavior on the bands I_l; $l = 0, \ldots, n$, so we define

$$G_{\mathbf{r}_i,l}(\omega) = \begin{cases} G_{\mathbf{r}_i}(\omega) & \text{if } \omega \in I_l \\ \text{"Don't care"} & \text{Otherwise} \end{cases} \tag{20}$$

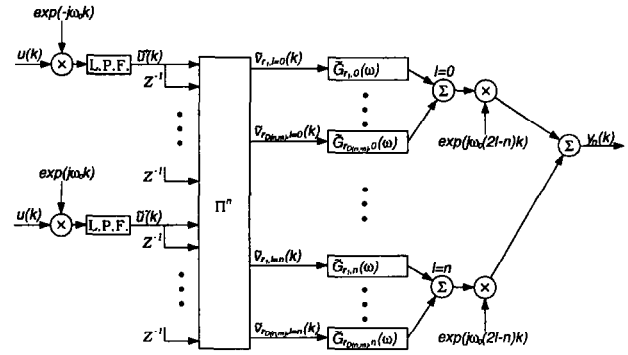

Figure 2: Frequency band decomposition implementation.

We may now express the output of the system in terms of output associated with each frequency band

$$Y_n(\omega) = \sum_{l=0}^{n} Y_{n,l}(\omega) \tag{21}$$

where

$$Y_{n,l}(\omega) = \sum_{i=1}^{D(n,m)} V_{\mathbf{r}_i,l}(\omega) G_{\mathbf{r}_i,l}(\omega) \tag{22}$$

This decomposition explicitly indicates the effect of the Volterra kernel on each frequency component of the output. This representation is particularly useful where the nonlinear effects on a limited number of frequency bands are of interest, since then only a subset of the $Y_{n,l}(\omega)$ need be evaluated. For example, in a communications system application the nonlinear terms that occur in the vicinity of the carrier frequency are of greatest concern, since the other nonlinear terms can be eliminated by linear filtering. This decomposition also suggests an efficient implementation for the Volterra filter.

3.2. Efficient Implementation via down-sampling

First note that the determination of $V_{\mathbf{r}_i,l}(\omega)$ and filtering by $G_{\mathbf{r}_i,l}(\omega)$ may be performed in terms of baseband data by frequency shifting the $U^+_{r_j}(\omega)$ and $U^-_{r_j}(\omega)$ to center them on DC. Denote the corresponding baseband time signals as $\tilde{u}^+_{r_j}(k)$ and $\tilde{u}^-_{r_j}(k)$. By multiplying the appropriate set of time signals $\tilde{u}^+_{r_j}(k)$ and $\tilde{u}^-_{r_j}(k)$, we obtain a baseband version of $v_{\mathbf{r}_i,l}(k)$, denoted as $\tilde{v}_{\mathbf{r}_i,l}(k)$. Next, filter $\tilde{v}_{\mathbf{r}_i,l}(k)$ with $\tilde{g}_{\mathbf{r}_i,l}(k)$, where $\tilde{g}_{\mathbf{r}_i,l}(k) = \exp(-j(2l-n)\omega_0 k) g_{\mathbf{r}_i,l}(k)$, to obtain $\tilde{y}_{n,l}(k)$. Lastly, we obtain $y_{n,l}(k)$ by modulating $\tilde{y}_{n,l}(k)$ to the l^{th} frequency band. That is,

$$y_{n,l}(k) = \exp(j(2l-n)\omega_0 k)\tilde{y}_{n,l}(k) \tag{23}$$

The baseband implementation of the Volterra filter is depicted in Figure 2. In summary, first the input is demodulated, then a baseband version of the nonlinear system associated with each frequency band is implemented and these outputs are modulated back to the proper place in the spectrum before combining them.

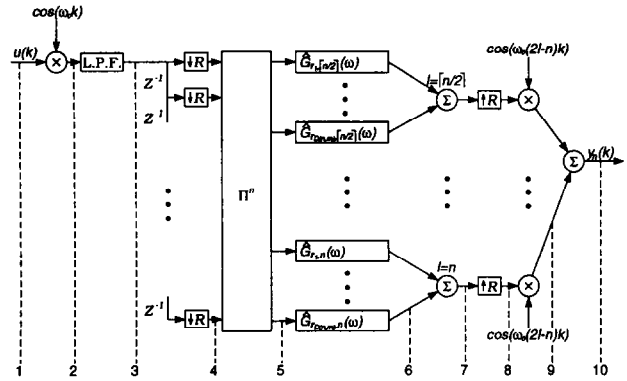

Figure 3: Efficient implementation via down-sampling.

The highest frequency component of the baseband data $\tilde{v}_{\mathbf{r}_i,l}(k)$ is at $\frac{n\Delta\omega}{2}$. If $\frac{n\Delta\omega}{2} \ll \omega_0$, then an efficient implementation is obtained by down-sampling $\tilde{u}^+(k)$ and $\tilde{u}^-(k)$ prior to computing the $\tilde{v}_{\mathbf{r}_i,l}(k)$. By so doing, both the multiplications required to compute $\tilde{v}_{\mathbf{r}_i,l}(k)$ and the filtering by $\tilde{g}_{\mathbf{r}_i,l}(k)$ are performed at a lower data rate. Let R be the down-sampling factor. We introduce the following notation for the down-sampled data

$$\hat{v}_{\mathbf{r}_i,l}(k) = \tilde{v}_{\mathbf{r}_i,l}(kR) \tag{24}$$

and down-sampled impulse response

$$\hat{g}_{\mathbf{r}_i,l}(k) = \tilde{g}_{\mathbf{r}_i,l}(kR) \tag{25}$$

The frequency domain representation of (24) and (25) are respectively

$$\hat{V}_{\mathbf{r}_i,l}(\omega) = \tilde{V}_{\mathbf{r}_i,l}(\omega/R) = V_{\mathbf{r}_i,l}(\omega/R + (2l-n)\omega_0) \tag{26}$$

and

$$\hat{G}_{\mathbf{r}_i,l}(\omega) = \tilde{G}_{\mathbf{r}_i,l}(\omega/R) = G_{\mathbf{r}_i,l}(\omega/R + (2l-n)\omega_0) \tag{27}$$

Additional efficiencies are obtained if $u(k)$ is real since then $U^+(\omega) = (U^-(\omega))^*$ [3]. In this case have $V_{\mathbf{r}_i,l}(\omega) = (V_{\mathbf{r}_i,n-l}(\omega))^*$ and it suffices to implement only one half of the frequency bands, that is, we require calculation for only $N_{\text{bands}} = \left\lceil \frac{n+1}{2} \right\rceil$ frequency bands instead of $n+1$ bands. Further efficiency is obtained if $u(k) = x(k)\cos(\omega_0 k)$ where $x(k)$ is real, since then $\tilde{u}^+(k) = \tilde{u}^-(k)$. Note that this implies $\tilde{v}_{\mathbf{r}_j,l}(k)$, $l = \left\lceil \frac{n+1}{2} \right\rceil, \ldots, n-1$ is a scalar multiple of $\tilde{v}_{\mathbf{r}_j,n}(k)$, so only $\hat{v}_{\mathbf{r}_j,n}(k)$ needs to be calculated. Figure 3 depicts a down-sampled, baseband implementation for this case. The steps labeled in Figure 3 are described as follows: In step 2 the input is frequency shifted and then lowpass filtered in step 3 to obtain a baseband signal corresponding to the positive frequency component of the input. Each of the time shifted baseband signals is downsampled in step 4 and products of the downsampled signals formed to obtain the $D(n,m)$ $\hat{v}_{\mathbf{r}_j,n}(k)$'s. Each $\hat{v}_{\mathbf{r}_j,n}(k)$ is then split into N_{bands} and multiplied by the corresponding scale factor before being filtered by the frequency shifted down-sampled impulse responses $\hat{g}_{\mathbf{r}_i,l}(k)$ in step 6. Next, we sum the $D(n,m)$ data streams belonging to each frequency band and up-sample (including interpolation) each of the N_{bands} bands. Lastly, in step 9 each band is moved it into its proper position in the spectrum, and the bands summed to obtain the overall output.

3.3. Computational Complexity of Down-Sampled Implementation

The reduction in computation afforded by down-sampling is obviously a function of the relative bandwidth, $\frac{\Delta\omega}{\omega_0}$. Clearly, there is a relative bandwidth for which the savings incurred by the lower sampling rate outweigh the overhead needed to down-sample, up-sample and separate into different frequency bands. For purposes of comparison we consider the case where $u(k) = x(k)\cos(\omega_0 k + \phi)$ with $x(k)$ real. It is straight forward to generalize these results to complex valued $x(k)$ or $u(k)$ although somewhat more tedious. We use real multiplications per sample as the standard of computational complexity. The computational complexity is not only a function of the kernel parameters and relative bandwidth, but also depends on the length of the low pass and interpolation filters. Whenever a generic low pass filter is needed we assume an equiripple linear phase FIR filter with a stop-band ripple $\delta_1 = 0.01$ and passband ripple $\delta_2 = \delta_1$. This gives 40 dB of stop-band attenuation. The filter length is estimated using Bellanger's formula as $\frac{2\log_{10}(\frac{1}{\delta_1\delta_2})}{3\Delta f}$, where $\Delta f = \frac{\omega_{stop}-\omega_{pass}}{2\pi}$. As before, we assume $2\pi = n(2\omega_0+\Delta\omega)$.

After splitting into frequency bands and down-sampling, the length of each $g_{\mathbf{r}_i,l}(k)$ should decrease, on average, by a factor of R. Let $\hat{N}_{\text{length}}$ denote the average filter length of $\hat{g}_{\mathbf{r}_i,l}(k)$. We shall bound $\hat{N}_{\text{length}}$ as $\max\{1, N_{\text{length}}/R\} \simeq \hat{N}_{\text{length}} < N_{\text{length}}$.

We now proceed counting the multiplies associated with each of the stages defined in the previous subsection. One multiplication per sample is necessary in step 2. In step 3 we implement a generic low pass filter with $\omega_{pass} = \frac{\Delta\omega}{2}$, $\omega_{stop} = 2\omega_0 - \frac{\Delta\omega}{2}$. This implies the filter length is $2n(2\omega_0 + \Delta\omega)/(2\omega_0 - \Delta\omega)$. Assuming $\Delta\omega \ll \omega_0$ gives a filter length of approximately $2n$. Hence, implementing this filter in the time domain requires $2n$ real multiplications. Note that the output of this stage is $x(k)$, which is real. At stage 4 we assume that R is an integer and thus no multiplications are required at this stage. Notice that from here until stage 8 the computations are performed at the lower sampling rate. In stage 5 we must generate $D(n,m)$ output streams $\hat{v}_{\mathbf{r}_i,n}(k)$. The number of multiplications per sample required to calculate these n^{th} order products is shown to be approximately $2D(n,m)$ in [3]. Hence, $2D(n,m)/R$ multiplications are required per output sample. Next we implement $N_{\text{bands}}D(n,m)$ filters of average length $\hat{N}_{\text{length}}$. Hence, there are $\frac{D(n,m)N_{\text{bands}}\hat{N}_{\text{length}}}{R}$ multiplications required per output sample at stage 6.

There are N_{bands} up-samples at stage 8 that each require an interpolation filter. Assuming a generic low pass filter with $\omega_{pass} = n\frac{\Delta\omega}{2}$, $\omega_{stop} = \frac{2\pi}{R} - n\frac{\Delta\omega}{2}$ we obtain a filter length of $\frac{2\log_{10}10^5}{3\Delta f} = \frac{20\pi R}{3(2\pi - n\Delta\omega R)}$. Thus, there are $N_{\text{bands}}\frac{20\pi R}{3(2\pi-n\Delta\omega R)}$ multiplications per sample. Note that there is a tradeoff between interpolation filter length and the down-sampling rate. As R increases, the required in-

terpolation filter length increases. It is convenient to express this tradeoff in a slightly different form. Clearly, $R < \frac{2\pi}{n\Delta\omega}$ so let $R = K\frac{2\pi}{n\Delta\omega}$ where $0 < K < 1$. Since $2\pi = n(2\omega_0 + \Delta\omega)$, we have $R = K\frac{2\omega_0+\Delta\omega}{\Delta\omega}$. Let $Q = \frac{\Delta\omega}{2\omega_0+\Delta\omega}$ denote the baseband system fractional bandwidth so that $R = \frac{K}{Q}$. Thus, the interpolation filter length is $\frac{10K}{3(1-K)Q}$. The number of real multiplications required for this stage is $N_{\text{bands}}\frac{10K}{3(1-K)Q}$. Lastly, there are N_{bands} multiplications per sample associated with stage 9.

Thus, combining each stage, the total number of multiplications per sample for the down-sampled implementation is

$$\begin{aligned} M_{\text{down}} &= 1 + 2n + \frac{D(n,m)Q}{K}(2 + N_{\text{bands}}\hat{N}_{\text{length}}) \\ &\quad + N_{\text{bands}}\frac{10K}{3(1-K)Q} + N_{\text{bands}} \end{aligned} \tag{28}$$

We now determine the K that minimizes the number of multiplications by setting the derivative of (28) with respect to K to zero. Define $P = \sqrt{\frac{10N_{\text{bands}}}{3D(n,m)(2+N_{\text{bands}}\hat{N}_{\text{length}})}}$. Hence, the minimal number of multiplications per sample is

$$\begin{aligned} M_{\text{down}} &= 1 + 2n + D(n,m)(Q+P)(2 + N_{\text{bands}}\hat{N}_{\text{length}}) \\ &\quad + N_{\text{bands}}\left(\frac{10}{3P} + 1\right) \end{aligned} \tag{29}$$

Recall that the number of multiplications per sample for a traditional serial implementation without down-sampling is $M_{\text{serial}} = D(n,m)(2 + N_{\text{length}})$. Thus, the ratio of down-sampling to traditional implementation multiplications is

$$\begin{aligned} \frac{M_{\text{down}}}{M_{\text{serial}}} &= \big(1 + 2n + D(n,m)(Q+P)(2 + N_{\text{bands}}\hat{N}_{\text{length}}) \\ &\quad + N_{\text{bands}}\,(10/3P + 1)\big) \\ &\quad /\,(D(n,m)\,(2 + N_{\text{length}})) \end{aligned} \tag{30}$$

We may obtain an upper bound on $\frac{M_{\text{down}}}{M_{\text{serial}}}$ by assuming $\hat{N}_{\text{length}} = N_{\text{length}}$, and a conservative lower bound by setting $\hat{N}_{\text{length}} = \max\{1, N_{\text{length}}/R_{max}\} = \max\{1, N_{\text{length}}Q\}$. This lower bound is conservative since we assumed $\hat{N}_{\text{length}}$ was independent of K when finding the optimum K.

It is difficult to intuitively assess the relative computational complexity directly from (30), so we offer representative examples in which the relative computational complexity is evaluated numerically. Figures 4 a) and b) depict the upper and lower bounds on $\frac{M_{\text{down}}}{M_{\text{serial}}}$ for a third and fifth order kernel, respectively, as a function of memory m assuming several different fractional bandwidths Q. As expected, the relative advantage of the down-sampling implementation increases as both the memory increases and the fractional bandwidth decreases. The down-sampling implementation is always advantageous for fractional bandwidths less than one-half and modest memory lengths. Note that a greater complexity reduction is obtained with the fifth order system than the third order system with small fractional bandwidths.

4. CONCLUSIONS

A diagonal coordinate representation for Volterra filters is exploited to develop efficient Volterra filter implementations for processing carrier based input signals. The diagonal coordinate representation expresses the output as a sum of linear filters applied to modified input sequences. This linear relationship illustrates the relationship between the kernels and the output spectrum. It also allows use of standard linear filtering techniques. The efficient implementation of systems with carrier based inputs are obtained by performing filtering on baseband signals. The relative computational complexity of the baseband implementation is proportional to the signal's fractional bandwidth.

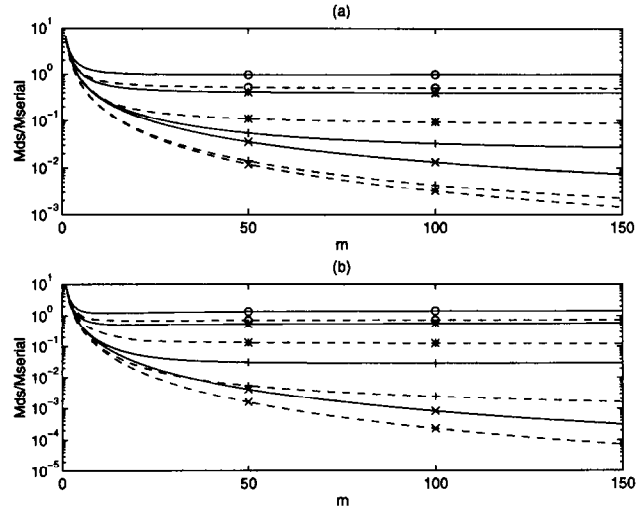

Figure 4: Ratio of number of multiplications of down-sampled implementation over serial implementation as function of m for several values of Q.
o - $Q = \frac{1}{2}$. * - $Q = \frac{1}{5}$. + - $Q = \frac{1}{100}$. × - $Q = \frac{1}{1000000}$.
Solid line - Upper bound. Dashed line - Lower bound.
a) $n = 3$. b) $n = 5$.

A. REFERENCES

[1] G. Lazzarin, S. Pupolin, and A. Sarti, "Nonlinearity compensation in digital radio systems," *IEEE Trans. Comm.*, vol. 42, pp. 988-998, 1994.

[2] J. C. Peyton and S. A. Billings, "Describing functions, Volterra series, and the analysis of nonlinear systems in frequency domain" *Int. J. Control*, 1991, Vol. 53, no. 4, pp. 871-887.

[3] G. M. Raz, B. D. Van Veen, "Baseband Volterra filters for implementing carrier based nonlinearities," *IEEE Trans. Signal Proc.*, Submitted, 1997.

[4] A.A.M. Saleh and J. Salz, "Adaptive linearization of power amplifiers in digital radio systems," *Bell Syst. Tech. J.*, vol. 62, pp. 1019-1033, 1983.

[5] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*. New York: Wiley, 1980.