ACCELERATED CONVERGENCE OF AN ITERATIVE IMPLEMENTATION OF A TWO-DIMENSIONAL IIR FILTER

Zhi Li and Robert J. Marks II
Interactive Systems Design Lab
Department of Electrical Engineering, FT-10
University of Washington
Seattle, WA 98195
(206)543-6990,543-6061 or 543-2150

ABSTRACT:

We investigate acceleration of the iterative implementation of the two-dimensional (2-D) infinite impulse response (IIR) filter proposed by Dudgeon [1-3]. The conventional procedure results in a sequence of estimations that converge linearly to the desired filter output. Following the recently proposed accelerated convergence algorithm developed for linearly distorted signal reconstruction problems [4], we use a product expansion of the denominator of the transfer function of a IIR filter to derive a class of iterative algorithms that have a p^{th} order rate of convergence. Specifically, k iterations of the proposed p^{th} order algorithm are equal to the p^k iterations of the linear algorithm. Therefore, the number of iterations required to obtain a given approximation is reduced from p^k to k. We analyze the error due to the spatial truncation in the implementation and conclude that the truncation error does not affect the computation of the desired result in the region of interest provided that the boundary values outside the area of interest are known. Finally, we present two examples to compare the difference in the convergence rates for the linear and the accelerated algorithms.

I. Introduction

The iterative implementation of a two-dimensional (2-D) IIR filter has been proposed by Dudgeon and his colleagues [1-3]. They have shown that a stable 2-D rational frequency response can be implemented by an iterative computation involving only finite-extent impulse response (FIR) filtering operations. This iterative implementation can be realized with digital processors which can convolve a 2-D signal with a filter kernel of limited extent very effectively. Furthermore, because the computation involves only an FIR filtering operation each iteration, the implementation can be used to approximate a rational frequency response which is not recursively computable, without the necessity of 2-D spectral factorization.

Even though this iterative implementation has been successfully applied, its convergence rate is relatively slow since the conventional procedure results in a sequence of estimations that converges linearly to the desired filter output. In this paper, we investigate the convergence acceleration for this iterative implementation following the recently proposed accelerated convergence algorithm developed for linearly distorted signal reconstruction problems [4]. In section II, we review the iterative implementation of a 2-D IIR filter. In section III, we derive a class of iterative algorithm that have a p^{th} order rate of convergence. We show that k iterations of the proposed p^{th} order algorithm are equivalent to p^k iterations of the linear algorithm. In section IV, we consider the computational efficiency based on the number of arithmetic operations and show that the accelerated algorithms use less multiplications and additions. In section V, we analyze the error due to spatial truncation in a practical implementation and conclude that this error does not effect the computation of the desired result in the region of interest provided that the boundary values outside the area of interest are known. Finally, in section V, we present two examples to compare the difference in the convergence rates of the linear and the proposed algorithm. One example is implementation of a 2-D low pass filter and the other is application of known boundary values to obtain the correct solution in a spatially truncated situation.

II. Dudgeon's IIR Filter Algorithm

A 2-D rational frequency response can be written as

$$H(\omega_1,\omega_2) = A(\omega_1,\omega_2)/B(\omega_1,\omega_2) \quad (1)$$

where $A(\omega_1,\omega_2)$ and $B(\omega_1,\omega_2)$ are trigonometric polynomials given by a 2-D discrete Fourier transformation (DFT) of arrays a(n,m) and b(n,m), respectively. Both a(n,m) and b(n,m)are assumed to be of finite extent.

We can define:

$$C(\omega_1,\omega_2) = 1 - B(\omega_1,\omega_2) \quad (2)$$

Therefore,

$$H(\omega_1,\omega_2) = A(\omega_1,\omega_2) /[1-C(\omega_1,\omega_2)] \quad (3)$$

If we denote x(n,m) and y(n,m) as the filter's input signal and output signal, respectively, and their DFT's are $X(\omega_1,\omega_2)$ and $Y(\omega_1,\omega_2)$, then, from (3), we have

ISCAS '89

$$Y(\omega_1,\omega_2) = A(\omega_1,\omega_2)X(\omega_1,\omega_2)/[1-C(\omega_1,\omega_2)]. \quad (4)$$

If $|C(\omega_1,\omega_2)| < 1$, the $Y(\omega_1,\omega_2)$ can be written as

$$Y(\omega_1,\omega_2) = A(\omega_1,\omega_2)X(\omega_1,\omega_2)\sum_{n=0}^{\infty} C^n(\omega_1,\omega_2). \quad (5)$$

If we denote

$$Y_k(\omega_1,\omega_2) = A(\omega_1,\omega_2)X(\omega_1,\omega_2)\sum_{n=0}^{k-1} C^n(\omega_1,\omega_2) \quad (6)$$

as the k^{th} approximation of $Y(\omega_1,\omega_2)$, we obtain the following iteration formula:

$$Y_{k+1}(\omega_1,\omega_2) = A(\omega_1,\omega_2)X(\omega_1,\omega_2) + C(\omega_1,\omega_2)Y_k(\omega_1,\omega_2) \quad (7a)$$

with

$$Y_0(\omega_1,\omega_2) = 0. \quad (7b)$$

This algorithm has been extended to implement an arbitrary stable rational filter [2,3]. Because of space limitations, the extension will not be included in this paper.

If we define $e_k(\omega_1,\omega_2)$ as the error after k iterations of (7) as

$$|e_k(\omega_1,\omega_2)| = |Y(\omega_1,\omega_2) - Y_k(\omega_1,\omega_2)| \quad (8)$$

Then, from (4) and (6),

$$|e_k(\omega_1,\omega_2)| = |Y(\omega_1,\omega_2)|\,|C(\omega_1,\omega_2)|^k \quad (9)$$

and the normalized error is

$$\xi_k(\omega_1,\omega_2) = |e_k(\omega_1,\omega_2)|/|Y(\omega_1,\omega_2)| = |C(\omega_1,\omega_2)|^k. \quad (10)$$

Since $|C(\omega_1,\omega_2)| < 1$, we see that the $\{\xi_k\}$ is a sequence that converges linearly to zero. As a result, we refer to (7) as the linear or first-order iteration algorithm.

III. Derivation of the Accelerated Algorithm

In this section, we derive a general class of iterative algorithms for the implementation of IIR filters. We use a product expansion of the denominator to obtain (1) the quadratic convergence algorithm and (2) the generalized p^{th} order convergence algorithm. In this paper, we only discuss the accelerated algorithm for (7). This acceleration method, however, can be modified to accommodate the extended iterative algorithm for an arbitrary stable filter.

A. The Iterative Algorithm with Quadratic Convergence

The linear iterative algorithm in (7) corresponds to a term-by-term implementation of the geometric series expansion of $1/B(\omega_1,\omega_2)$:

$$1/B(\omega_1,\omega_2) = 1/[1-C(\omega_1,\omega_2)] = 1+C(\omega_1,\omega_2)+C^2(\omega_1,\omega_2)+... \quad (11)$$

If we separate the sum of (11) into two summations, one containing the even power terms of $C(\omega_1,\omega_2)$and other the odd power terms, we obtain

$$1/B(\omega_1,\omega_2) = \sum_{n=0,2,4} C^n(\omega_1,\omega_2) + \sum_{n=1,3,5} C^n(\omega_1,\omega_2) = [1+C(\omega_1,\omega_2)]\sum_{m=0}^{\infty} C^{2m}(\omega_1,\omega_2) \quad (12)$$

The m sum is divided again into even and odd components and the process is repeated. Repetitive application of this procedure results in

$$1/B(\omega_1,\omega_2) = \prod_{n=0}^{\infty}[1+C^{2^n}(\omega_1,\omega_2)] \quad (13)$$

Applying this product expansion to (4), we get

$$Y(\omega_1,\omega_2) = A(\omega_1,\omega_2)X(\omega_1,\omega_2)\prod_{n=0}^{\infty}[1+C^{2^n}(\omega_1,\omega_2)] \quad (14)$$

From (14), we would like to derive an iterative algorithm. Let

$$Y_k(\omega_1,\omega_2) = A(\omega_1,\omega_2)X(\omega_1,\omega_2)\prod_{n=0}^{k-1}[1+C^{2^n}(\omega_1,\omega_2)] \quad (15)$$

be the k^{th} estimation. Then $Y_{k+1}(\omega_1,\omega_2)$ is related to $Y_k(\omega_1,\omega_2)$ by

$$Y_{k+1}(\omega_1,\omega_2) = [1 + C^{2^k}(\omega_1,\omega_2)]Y_k(\omega_1,\omega_2) = Y_k(\omega_1,\omega_2)+T_k(\omega_1,\omega_2)Y_k(w_1,\omega_2), \quad (16a)$$

with

$$Y_0(\omega_1,\omega_2) = A(\omega_1,\omega_2)X(\omega_1,\omega_2), \quad (16b)$$

where we have defined

$$T_k(\omega_1,\omega_2) = C^{2^k}(\omega_1,\omega_2) \quad (17)$$

Note that $T_k(\omega_1,\omega_2)$ may be recursively updated as:

$$T_k(\omega_1,\omega_2) = T^2_{k-1}(\omega_1,\omega_2). \quad (18)$$

Equations (16) and (18) together constitute our iterative algorithm with quadratic convergence. The algorithm is illustrated in Fig.1.

In order to study the convergence rate of the iterative algorithm in (16), we expand the product factorization for $Y_k(\omega_1,\omega_2)$ in (15) and we get, for $k > 0$,

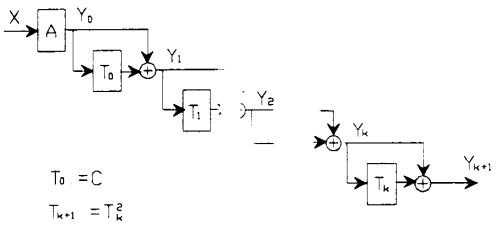

FIG.1 BLOCK DIAGRAM FOR THE QUADRATIC ALGORITHM

$$Y_k(\omega_1,\omega_2) = A(\omega_1,\omega_2)X(\omega_1,\omega_2)\sum_{n=0}^{2^k-1} C^n(\omega_1,\omega_2) \quad (19)$$

Comparing (19) with (6), we observe that k iterations of (16) are equivalent to 2^k iterations of the linear algorithm (7). Proceeding as we did for the linear algorithm, we can analyze the error after k iterations. From (19):

$$|e_k(\omega_1,\omega_2)| = |Y_k(\omega_1,\omega_2) - Y(\omega_1,\omega_2)| = |Y(\omega_1,\omega_2)C^{2^k}(\omega_1,\omega_2)|. \quad (20)$$

The normalized error follows as

$$\xi_k(\omega_1,\omega_2) = |e_k(\omega_1,\omega_2)| / |Y(\omega_1,\omega_2)|$$
$$= |C(\omega_1,\omega_2)|^{2^k}. \quad (21)$$

If $|C(\omega_1,\omega_2)|<1$, then $\{\xi_k\}$ is a sequence that converges quadratically to zero. Therefore, we refer to this algorithm as the quadratic or second-order iterative algorithm.

In any practical implementation, the number of iterations is finite. We can take the normalized error as one measurement of the output spectral error introduced by terminating the iterative computation after k iterations. If we specify a tolerable degree of spectral error of $\varepsilon > 0$,

$$\xi_k(\omega_1,\omega_2) = |C(\omega_1,\omega_2)|^{2^k} < \varepsilon, \quad (22)$$

then we can tell how many iterations will be needed for a given $C(\omega_1,\omega_2)$.

B. An Iterative Algorithm with pth Order Convergence

Generally, we can derive a p^{th} order iterative algorithm. In deriving the quadratic algorithm, we separated the geometric series expansion of $1/B(\omega_1,\omega_2)$ into even and odd powers of $C(\omega_1,\omega_2)$ as one does for a radix-2 decimation-in-time FFT. In the same way, if we decimate the series in (11) with a radix of p, we can get the following p^{th} order factorization:

$$1/B(\omega_1,\omega_2) = \sum_{m=0}^{\infty} C^n(\omega_1,\omega_2)$$
$$= \prod_{n=0}^{\infty} [1+C^{p^n}+C^{2p^n}+\ldots+C^{(p-1)p^n}]. \quad (23)$$

The corresponding k^{th} estimation $Y_k(\omega_1,\omega_2)$ is

$$Y_k(\omega_1,\omega_2) = A(\omega_1,\omega_2)\, X(\omega_1,\omega_2) \prod_{n=0}^{k-1} [1+C^{p^n}(\omega_1,\omega_2)$$
$$+C^{2p^n}(\omega_1,\omega_2)+\ldots + C^{(p-1)p^n}(\omega_1,\omega_2)]. \quad (24)$$

The corresponding iterative algorithm is

$$Y_{k+1}(\omega_1,\omega_2) = Y_k(\omega_1,\omega_2)\,[1+\sum_{m=1}^{p-1} T^m{}_k(\omega_1,\omega_2)] \quad (25a)$$
$$Y_0(\omega_1,\omega_2) = A(\omega_1,\omega_2)\, X(\omega_1,\omega_2) \quad (25b)$$
$$T_{k+1}(\omega_1,\omega_2) = T_k^p(\omega_1,\omega_2) \quad (25c)$$

and

$$T_0(\omega_1,\omega_2) = C(\omega_1,\omega_2). \quad (25d)$$

From the equation

$$\prod_{n=0}^{k-1} (1+C^{p^n}+C^{2p^n}+\ldots+C^{(p-1)p^n}) = \sum_{n=0}^{p^k-1} C^n, \quad (26)$$

it is clear that k iterations of (25) are equivalent to p^k iterations of the linear algorithm.

We can easily show the normalized error is

$$\xi_k = |C(\omega_1,\omega_2)|^{p^k} \quad (27)$$

Just as we did in (22), we can use (27) to determine the number of iterations for this p^{th} order iteration to converge to a given tolerable spectrum error.

IV. Spatial Truncation Error Analysis

In this section, we will discuss one kind of error in the practical implementation due to spatial truncation dictated by a finite capacity memory. Although we only consider the quadratic algorithm, the analysis can be extended to the p^{th} order algorithm.

If the quadratic algorithm in (16) is implemented in the spatial domain with a digital processor, we have:

$$y_{k+1}(n,m) = y_k(n,m)+h_k(n,m)**y_k(n,m) \quad (28a)$$

$$y_0(n,m) = a(n,m) ** x(n,m) \quad (28b)$$

where we have defined:

$$h_k(n,m) = \text{DFT-1}[C^{2^k}(\omega_1,\omega_2)] \quad (29)$$

and ** means the 2-D convolution. As the iteration continues, the dispersion of $y_{k+1}(n,m)$ will increase and eventually exceed the capacity of available memory.

In order to analyze this truncation error, let us define a truncation operator T by

$$T[q(n,m)] = \begin{cases} q(n,m); & (n,m) \in I \\ 0; & (n,m) \notin I \end{cases} \quad (30)$$

where I is a region of the interest and q(n,m) is any 2-D signal.

Imposing T onto (28a), we obtain the truncated version of

$y_k(n,m)$, denoted by $\bar{y}(n,m)$,

$$\bar{y}_{k+1}(n,m) = T[\bar{y}_k(n,m)+h_k(n,m)**\bar{y}_k(n,m)] \quad (31)$$

We can define the truncation error as

$$e_k(n,m) = y_k(n,m) - \bar{y}_k(n,m); \quad (n,m) \in I \quad (32)$$

From (28) and (32), we get

$$T[e_{k+1}(n,m)] = T[e_k(n,m) + h_k(n,m)**e_k(n,m)] \quad (33)$$

That is, the truncation error can be computed iteratively with incorporation of the boundary condition:

$$e_{k+1}(n,m) = \begin{cases} e_k(n,m)+h_k(n,m)**e_k(n,m); & (n,m)\in I \\ y(n,m); & (n,m)\notin I \end{cases} \quad (34)$$

In the region outside I, the error signal is just y(n,m) because $\bar{y}(n,m)$ is zero. Thus

$$y(n,m) = \bar{y}(n,m) + e(n,m) \quad (35)$$

If we take the T operation to the both sides of (31) and add the result to (33), we can prove that y(n,m) is a solution to the iteration

$$y_{k+1}(n,m) = \begin{cases} y_k(n,m) + h_k(n,m)**y_k(n,m); & (n,m)\in I \quad (36a) \\ y(n,m); & (n,m)\notin I \quad (36b) \end{cases}$$

Therefore, even though the truncation operator will in general introduce some error, the correct signal can still be computed provided that the values at the boundary are known.

(217) 333-4789 Further information can be obtained from Dr. W. Kenneth Jenkins.

Note that in order to perform the 2-D convolution of (36a), the boundary area needs only to be big enough to cover the array $h_k(n,m)$ when $y_{k+1}(n,m)$ is being computed for (n,m) on the edge of the region I. From (29), it is evident that the spatial extent of $h_k(n,m)$ will change with the iteration index k. Furthermore, if a general p^{th} order algorithm is used, the extent of $h_k(n,m)$ will also change with p.

VI. Examples

In this section, we present two examples to compare the difference in the convergence rates for the linear and the accelerated algorithms. The first one is to implement a 2-D low pass filter with the transfer function:

$$H(\omega_1,\omega_2)= A(\omega_1,\omega_2)/[1-C(\omega_1,\omega_2)] \quad (37a)$$

where

$$A(\omega_1,\omega_2) = 0.50033+0.81561f +0.41543(2f^2 - 1)$$

$$+0.11311(4f^3-3f) -0.000037592(8f^4 -8f^2 +1); \quad (37b)$$

$$C(\omega_1,\omega_2) = -0.0010153f-0.83047(2f^2 -1); \quad (37c)$$

with

$$f=0.5[-1+\cos(\omega_1)+\cos(\omega_2)+\cos(\omega_1)\cos(\omega_2)] \quad (37d)$$

The second example illustrates use of the boundary condition to mitigate the effect of spatial truncation.

Dudgeon [1] implements the same filter of (37) using more than 20 iterations to reach a convergent solution. We realize this filter using only 5 iterations of quadratic algorithm and 3 iterations of cubic (i.e, p=3 in (25)) algorithm, respectively. In each case, we obtain results that are graphically indistinguishable from the magnitude response shown in Fig.2.

Note that in this example, 5 iterations of the quadratic algorithm are equivalent to $2^5 = 32$ iterations of the linear one, and 3 iterations of the cubic method are the same as $3^3 = 27$ iterations of the linear one.

For the second example, we use the quadratic algorithm to implement the example on p. 232 of [3]:

$$y_k(n)=0.6\delta(n)+0.4y_{k-1}(n-1)+0.4y_{k-1}(n+1); \quad (38)$$

which has the solution

$$y(n) = 0.5^{|n|} \quad (39)$$

Now, suppose we truncate the iteration as

$$\bar{y}_k(n)=$$

$$\begin{cases} 0.6\delta(n)+0.4\bar{y}_{k-1}(n-1)+0.4\bar{y}_{k-1}(n+1); & |n|<4 \\ 0; & |n| \geq 4 \end{cases} \quad (40)$$

Fig.3 shows the $\bar{y}(n)$ obtained by iterating (40) 5 times with the quadratic algorithm. However, we can find the correct y(n) by using the correct boundary conditions in the region of $|n| \geq 4$: $y(n)=0.5^{|n|}$. Fig.4 shows convergence after 5 iterations when the quadratic algorithm is used. In Ref. [3], 20 iterations are needed to obtain a similar result.

VII. Conclusions

In this paper, we have presented a class of algorithms with accelerated convergence for an iterative implementation of a 2-D IIR filter. Compared with the conventional linear algorithm, the proposed p^{th} order algorithm can reduce the number of iterations from p^k to k for a given tolerable spectrum error. A more significant comparison, of courses, is the relative number of operations and memory requirements for each algorithm. At this writing, this study remains undone. The error introduced by spatial truncation due to limited memory space was addressed. Also, we presented two examples to compare the different convergence rates for different algorithms. The examples show that the accelerated algorithms can converge to the desired solution at higher rates.

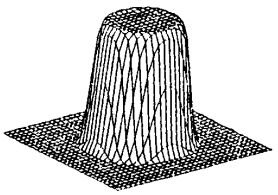

FIG.2 MAGNITUDE RESPONSE OF THE FILTER OF (37)

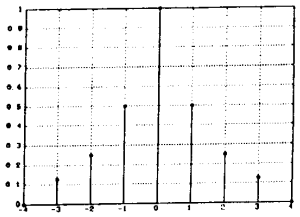

FIG.3 SOLUTION OF (40) BY 5 ITERATIONS OF QUADRATIC ALGORITHM

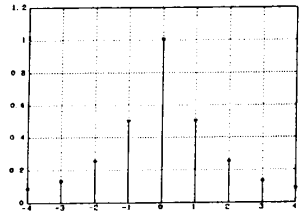

FIG.4 GENERATING THE CORRECT SOLUTION WITH THE APPROPRIATE BOUNDARY CONDITIONS WITH QUADRATIC ALGORITHM FOR 5 ITERATIONS

REFERENCES:

1. Dan E. Dudgeon, "An Iterative Implementation for 2-D Digital Filters," **IEEE Trans. Acoustics, Speech, and Signal Processing**, ASSP-28, no.6 (Dec.1980), 666 - 71.

2. Thomas F. Quatieri and Dan E. Dudgeon, "Implementation of 2-D Digital Filters by Iterative Methods," **IEEE Trans. Acoustics, Speech, and Signal Processing**, ASSP-30, no. 3,(June 1982), 473 - 87.

3. Dan E. Dudgeon and Russel M. Mersereau, **Multidimensional Digital Signal Processing** (Englewood cliffs, N.J.:Prentice-Hall,Inc.,1984)

4. Craig E. Morris, Mark A. Richards and Monson H. Hayes, "Fast Reconstruction of Linearly Distorted Signals," **IEEE Trans. Acoustics, Speech, and Signal Processing**, ASSP-36, no.7 (July 1988), 1017 - 25.