TWISTED FILTER BANKS FOR COMMUNICATION APPLICATIONS

Andreas Klappenecker
Texas A&M University
Department of Computer Science
College Station, TX 77843-3112, USA

ABSTRACT
We present a novel class of multirate filter banks that are based on a special class of time-varying filters. The filter operations are done by modifying the usual convolution operations with a twist. We will show that the filter banks allow perfect reconstruction, in spite of the fact that the filters are time-varying and the arithmetic is over finite fields. We prove that the one-dimensional twisted convolutions over finite fields are of a particularly simple form, which allows an efficient implementation.

1. INTRODUCTION
Many signal processing applications are based on convolution operations. For example, multirate filter banks found prominent applications in lossy compression of images. It is less known that multirate filter banks also occur in the context of communication applications. A striking example is given by an encoder of a convolutional code, which can be (and usually is) realized by a multirate filter bank over a finite field.

We discuss in this note a new class of multirate filter banks, which are based on twisted convolutions instead of traditional convolutions. With a view towards communication applications, we will discuss such filter banks for signals with values in a finite field \( F \).

Let \( f = (f_i)_{i \in \mathbb{Z}} \) and \( s = (s_j)_{j \in \mathbb{Z}} \) be two sequences with values in a finite field \( F \). Roughly speaking, the twisted convolution is given by the sequence

\[
(1)
\]

We can now restate the definition of a twisted convolution more precisely as follows:

**Definition 1** The operation \( \#: F[Z] \times F[Z] \rightarrow F[Z] \) defined by (1) is said to be a twisted convolution if and only if the function \( \omega \) satisfies the relations (2) and (3).
3. TWISTED FILTER BANKS

The basic principle of a twisted multirate filter bank is simple: subject the input signal \( s \) to \( n \) different twisted convolutions

\[ f_1 \# s, \ldots, f_n \# s, \]

and reduce the sampling rate of the resulting signals by keeping only every \( m \)th sample. Figure 1 shows the dataflow graph of such a twisted filter bank.

![Diagram of Analysis Filter Bank with Twisted Convolutions](image)

Figure 1. Analysis filter bank with twisted convolutions. This filter bank has \( n \) channels and reduces the sampling rate in each channel by \( m \).

Although twisted convolutions are in general not time-invariant, we will demonstrate in the following sections that they are nevertheless surprisingly well-behaved. For instance, we will obtain as a consequence that all one-dimensional twisted convolutions allow simple implementations. Another consequence will be a simple characterization of perfect reconstructing twisted filter banks.

4. EXAMPLES OF TWISTED CONVOLUTIONS

Definition 1 does not reveal much of the nature of twisted convolutions. Therefore, we pause here to give some simple examples. Surprisingly, it will turn out that these examples play a major role in the further development of the theory.

A twisted convolution (1) is determined by a function \( \omega: \mathbb{Z} \times \mathbb{Z} \to \mathbb{F}^\times \), which satisfies (2) and (3). Particularly simple examples are given by

\[ \omega(x, y) = \frac{\alpha(x)\alpha(y)}{\alpha(x + y)} \]

where \( \alpha: \mathbb{Z} \to \mathbb{F}^\times \) is a function with nonzero values in a finite field \( \mathbb{F} \) satisfying the normalization condition

\[ \alpha(0) = 1. \]

Indeed, the latter constraint ensures that \( \omega \) satisfies the normalization condition (2). It remains to check that \( \omega \) satisfies the condition (3). It follows from definition (4) that

\[ \omega(i, j)\omega(i + j, k) = \frac{\alpha(i)\alpha(j)\alpha(i + j)\alpha(k)}{\alpha(i + j)\alpha(i + j + k)} =: A, \]

and

\[ \omega(j, k)\omega(i, j + k) = \frac{\alpha(j)\alpha(k)\alpha(i)\alpha(j + k)}{\alpha(j + k)\alpha(i + j + k)} =: B. \]

Simplifying the previous two expressions \( A \) and \( B \) yields

\[ A = \frac{\alpha(i)\alpha(j)\alpha(k)}{\alpha(i + j + k)} = B, \]

which shows that condition (3) is indeed satisfied.

An appealing aspect of these examples is that they allow an efficient implementation. This can be achieved by a simple reduction of twisted convolutions to traditional convolutions, provided that \( \omega \) is of the form (4). This is accomplished at the expense of three additional multipliers and a generator for the sequences \((\alpha(n))\) and \((\alpha(n)^{-1})\), on top of the traditional convolution unit.

We can derive this implementation as follows. Assume that the input is given by \((s_j)_{j \in \mathbb{Z}}\), and the filter by \((f_j)_{j \in \mathbb{Z}}\). We multiply these sequences pointwise with the sequence \((\alpha(j))_{j \in \mathbb{Z}}\), and thus obtain

\[ (\alpha(j)s_j)_{j \in \mathbb{Z}} \quad \text{and} \quad (\alpha(j)f_j)_{j \in \mathbb{Z}}. \]

We feed these two sequences into a convolution unit. The output of this convolution unit is multiplied pointwise with the sequence \((\alpha(j)^{-1})_{j \in \mathbb{Z}}\). Therefore, we get as a result the sequence \((r_k)_{k \in \mathbb{Z}}\), where

\[ r_k = \sum_{j \in \mathbb{Z}} \frac{\alpha(k - j)\alpha(j)}{\alpha(k)} f_{k - j}s_j. \]

Substituting \( x = k - j \) and \( y = j \) in (4), we actually obtain

\[ r_k = \sum_{j \in \mathbb{Z}} \omega(k - j, j)f_{k - j}s_j. \]

Therefore, this scheme provides a simple implementation of the twisted convolution \( f \# s \), provided that \( \omega \) is of the form (4).

This prompts the natural question, how likely it is that \( \omega \) happens to be of the nice form (4)? We will answer this question in the following section.

5. FUNDAMENTAL PROPERTIES

Our definition of a twisted convolution ensured that \( \# \) is an associative operation, that is, the rule

\[ a \# (b \# c) = (a \# b) \# c \]
holds for all sequences \(a, b, c \in F[Z]\). In this section, we would like to show that \(\#\) is a commutative operation. In other words, we want to show that the rule

\[a \# b = b \# a\]

holds for all \(a, b \in F[Z]\). This is a rather surprising feature, which is peculiar to the one-dimensional setting.

It is easy to see that \(\#\) is commutative if and only if \(\omega(x, y) = \omega(y, x)\) holds for all \(x, y \in Z\). This property certainly holds for functions \(\omega\) of the form (4), since

\[\frac{\alpha(x)\alpha(y)}{\alpha(x + y)} = \frac{\alpha(y)\alpha(x)}{\alpha(y + x)}\]

is trivially satisfied for all \(x, y \in Z\). We will show that in fact all function \(\omega\) satisfying (2) and (3) are of the form (4).

We will use some cohomological methods to prove this result. We remark here that functions \(\omega: Z \times Z \to F^\times\) satisfying (2) and (3) are known as normalized 2-cocycles in cohomology theory. They have been studied by Schur, Hopf, and others in the early 20th century. A simple, but relevant result is that the normalized 2-cocycles form a group:

**Lemma 1** The set of all normalized 2-cocycles \(\omega\) from the integers with values in the group of units \(F^\times\) of the finite field \(F\) form an abelian group under pointwise multiplication. This group is denoted by \(Z^2(Z, F^\times)\).

**Proof.** Let \(\omega_1\) and \(\omega_2\) be normalized 2-cocycles. The product

\[\omega(x, y) = \omega_1(x, y)\omega_2(x, y)\]

is again a normalized 2-cocycle, since the relations (2) and (3) are a consequence of the relations for \(\omega_1\) and \(\omega_2\). The identity element of \(Z^2(Z, F^\times)\) is given by the constant 2-cocycle \(\omega(x, y) = 1\) for all \(x, y \in Z\). The inverse \(\omega^{-1}\) of an element \(\omega \in Z^2(Z, F^\times)\) is given by \(\omega^{-1}(x, y) = \omega(x, y)^{-1}\). This shows that \(Z^2(Z, F^\times)\) is indeed a group. \(\square\)

The normalized 2-cocycles of the form (4) are called coboundaries in cohomology theory. The coboundaries form a subgroup \(B^2(Z, F^\times)\) of the abelian group \(Z^2(Z, F^\times)\). Our quest is to show that all normalized 2-cocycles are actually coboundaries. For that purpose, we can study the factor group \(H^2(Z, F^\times) = Z^2(Z, F^\times)/B^2(Z, F^\times)\), which is known as the second cohomology group of \(Z\). If this group is trivial, then all normalized 2-cocycles are actually coboundaries.

**Theorem 2** Let \(F\) be a finite field. The second cohomology group \(H^2(Z, F^\times)\) is trivial, hence all normalized 2-cocycles with values in \(F^\times\) are in fact coboundaries.

**Proof.** We use some standard cohomological arguments to prove this result. A brief introduction to cohomology can be found for example in Curtis and Reiner [2, Chapter 8].

Let \(G\) be the infinite cyclic group with generator \(t\), i.e., \(G \cong Z\). A projective resolution of \(ZG\) over \(Z\) is given by

\[0 \to ZG \xrightarrow{\ell^{-1}} ZG \to Z \to 0\]

This shows that the cohomological dimension is

\[\text{cd } G = \text{proj dim}_{ZG} Z = 1.\]

Consequently, the second cohomology group is trivial

\[H^2(G, F^\times) = \text{Ext}^2_{ZG}(Z, F^\times) = 1,\]

see e.g. Brown [1, p. 184]. \(\square\)

This result shows in particular that all twisted convolutions \(\#\) can be efficiently implemented with the methods described in Section 4.

### 6. Perfect Reconstruction

In this section, we want to describe the conditions under which an \(n\)-channel twisted multirate filter bank allows perfect reconstruction of all input signals.

The analysis and synthesis filter banks work as follows. We assume that the input signal is subjected to \(n\) twisted convolutions \(f_k\), where \(1 \leq k \leq n\). The sampling rate of the output of these analysis filters is reduced by keeping every \(m\)th sample. On the synthesis side, the sampling rate is increased by inserting \(m - 1\) zeros between the samples. Finally, we apply twisted convolutions with synthesis filters \(g_k\) and add the results.

We will use the following notation. Signals and filters are expressed in terms of generating functions, as is explained in the Introduction. The twisted convolution \(f\#s\) of sequences \(f\) and \(s\) is expressed by the twisted product \(f(z)\#s(z)\). Thus, this twisted product \(\#\) of Laurent polynomials is defined by bilinear extension of the rule \(z^i\#z^j = \omega(i, j)z^{i+j}\). We note that the set of Laurent polynomials with coefficients in \(F\) equipped with addition \(+\) and twisted product \(\#\) forms a commutative ring, called the twisted polynomial ring; we denote this ring by \(F_\omega[z, z^{-1}]\).

We give now a formal description of the dataflow of an \(n\)-channel twisted filter bank using the notation introduced above. Suppose that we are given an input signal \(s(z)\). We form the twisted convolutions \(f_k(z)\#s(z), 1 \leq k \leq n\). We subject these signals to downsampling \([\downarrow m]\) and upsampling \([\uparrow m]\). Writing \(s(z)\) in the form

\[s_1(z^m) + z\#s_2(z^m) + \cdots + z^{m-1}\#s_m(z^m)\]

and expressing \(f_k(z)\) as

\[f_{k,1}(z^m) + z^{-1}\#f_{k,2}(z^m) + \cdots + z^{-(m-1)}\#f_{k,m}(z^m)\]

(6)
allows us to express $\Omega(z)$ as

$$\Omega(z) = \sum_{\ell=0}^{m-1} \omega(-\ell, \ell) f_{k, \ell+1}(z^m) \# s_{\ell+1}(z^m). \tag{7}$$

The intermediate signal $\Omega(z)$ is then processed further by twisted convolution with the synthesis filter $g_k(z)$. The output of the filter bank $\hat{s}(z)$ is obtained by adding the output of the channels:

$$\hat{s}(z) := \sum_{k=1}^{n} g_k(z) \# \Omega_k(z). \tag{8}$$

A twisted filter bank is called perfect reconstructing if and only if the reconstructed output signal $\hat{s}(z)$ coincides with the input $s(z)$ for all $s(z) \in \mathbb{F}[z, z^{-1}]$.

The perfect reconstruction property is most easily understood in terms of matrices, which highlight the relation between analysis and synthesis filters. We have summarized all processing steps of a twisted filter bank in Figure 2. We will now explain these steps in some detail.

**Analysis.** The analysis filter bank has $n$ channels. The input is processed in each channel by a twisted convolution with an analysis filter, followed by a sampling reduction $\lceil \cdot \rceil$. Afterwards, an upsampling operation $\lceil \cdot \rceil$ is applied. The result of these three operations is

$$\Omega_k(z) = \lceil \cdot \rceil \lceil \cdot \rceil \lceil \cdot \rceil (f(z) \# s(z)).$$

The underbraced part on the right hand side of Figure 2 computes $[\Omega_1(z), \Omega_2(z), \ldots, \Omega_n(z)]^t$. Indeed, this is a consequence of equation (7).

**Synthesis.** The remaining part simply needs to realize the twisted convolutions with the synthesis filters $g_k(z)$ with the intermediate results $\Omega_k(z)$. Therefore, the underbraced part on the left in Figure 2 needs to realize the vector $[g_1(z), g_2(z), \ldots, g_n(z)]$.

Figure 2. Formal description of the analysis and synthesis steps of an $n$-channel twisted filter banks with uniform sampling rate reduction by a factor of $m$. The matrix entries should be interpreted as elements of the twisted Laurent polynomial ring $\mathbb{B}[\omega, \omega^{-1}]$, that is, the product of two matrix elements is always given by the twisted product. The matrix $D$ denotes the diagonal matrix $D = \text{diag}(\omega(0), \omega(-1, 1), \ldots, \omega(-m+1, 1))$. The perfect reconstruction property is most easily understood in terms of matrices, which highlight the relation between analysis and synthesis filters. We have summarized all processing steps of a twisted filter bank in Figure 2. We will now explain these steps in some detail.

Afterwards, an upsampling operation $\lceil \cdot \rceil$ is applied. The result of these three operations is

$$\Omega_k(z) = \lceil \cdot \rceil \lceil \cdot \rceil \lceil \cdot \rceil (f(z) \# s(z)).$$

The underbraced part on the right hand side of Figure 2 computes $[\Omega_1(z), \Omega_2(z), \ldots, \Omega_n(z)]^t$. Indeed, this is a consequence of equation (7).

The twisted filter bank with analysis filters $g_k(z)$ is a left inverse of $H$, $G^t H = I$.

7. CONCLUSIONS

We have demonstrated that it is possible to construct twisted filter banks over finite field that allow perfect reconstruction. We derived efficient implementations of twisted convolutions. An interesting aspect for further studies is the fact that it is possible to adaptively change from one twisted convolution to another by changing the sequence $\alpha$.

8. REFERENCES
