Convolution Algorithms*

C. Sidney Burrus

1 Fast Convolution by the FFT

One of the main applications of the FFT is to do convolution more efficiently than the direct calculation from the definition which is:

$$y(n) = \sum h(m)\ x(n-m) \tag{1}$$

which, with a change of variables, can also be written as:

$$y(n) = \sum x(m)\ h(n-m) \tag{2}$$

This is often used to filter a signal $x(n)$ with a filter whose impulse response is $h(n)$. Each output value $y(n)$ requires N multiplications and $N-1$ additions if $y(n)$ and $h(n)$ have N terms. So, for N output values, on the order of N^2 arithmetic operations are required.

Because the DFT converts convolution to multiplication:

$$DFT\{y(n)\} \ = \ DFT\{h(n)\}\ DFT\{x(n)\} \tag{3}$$

can be calculated with the FFT and bring the order of arithmetic operations down to $Nlog(N)$ which can be significant for large N.

This approach, which is called "fast convolutions", is a form of block processing since a whole block or segment of $x(n)$ must be available to calculate even one output value, $y(n)$. So, a time delay of one block length is always required. Another problem is the filtering use of convolution is usually non-cyclic and the convolution implemented with the DFT is cyclic. This is dealt with by appending zeros to $x(n)$ and $h(n)$ such that the output of the cyclic convolution gives one block of the output of the desired non-cyclic convolution.

For filtering and some other applications, one wants "on going" convolution where the filter response $h(n)$ may be finite in length or duration, but the input $x(n)$ is of arbitrary length. Two methods have traditionally used to break the input into blocks and use the FFT to convolve the block so that the output that would have been calculated by directly implementing (1) or (2) can be constructed efficiently. These are called "overlap-add" and "over-lap save".

*Version 1.10: Apr 10, 2010 1:46 pm -0500
†http://creativecommons.org/licenses/by/2.0/

1.1 Fast Convolution by Overlap-Add

In order to use the FFT to convolve (or filter) a long input sequence $x(n)$ with a finite length-M impulse response, $h(n)$, we partition the input sequence in segments or blocks of length L. Because convolution (or filtering) is linear, the output is a linear sum of the result of convolving the first block with $h(n)$ plus the result of convolving the second block with $h(n)$, plus the rest. Each of these block convolutions can be calculated by using the FFT. The output is the inverse FFT of the product of the FFT of $x(n)$ and the FFT of $h(n)$. Since the number of arithmetic operation to calculate the convolution directly is on the order of M^2 and, if done with the FFT, is on the order of $Mlog(M)$, there can be a great savings by using the FFT for large M.

The reason this procedure is not totally straightforward, is the length of the output of convolving a length-L block with a length-M filter is of length $L+M-1$. This means the output blocks cannot simply be concatenated but must be overlapped and added, hence the name for this algorithm is "Overlap-Add".

The second issue that must be taken into account is the fact that the overlap-add steps need non-cyclic convolution and convolution by the FFT is cyclic. This is easily handled by appending $L-1$ zeros to the impulse response and $M-1$ zeros to each input block so that all FFTs are of length $M+L-1$. This means there is no aliasing and the implemented cyclic convolution gives the same output as the desired non-cyclic convolution.

The savings in arithmetic can be considerable when implementing convolution or performing FIR digital filtering. However, there are two penalties. The use of blocks introduces a delay of one block length. None of the first block of output can be calculated until all of the first block of input is available. This is not a problem for "off line" or "batch" processing but can be serious for real-time processing. The second penalty is the memory required to store and process the blocks. The continuing reduction of memory cost often removes this problem.

The efficiency in terms of number of arithmetic operations per output point increases for large blocks because of the $Mlog(M)$ requirements of the FFT. However, the blocks become very large ($L > > M$), much of the input block will be the appended zeros and efficiency is lost. For any particular application, taking the particular filter and FFT algorithm being used and the particular hardware being used, a plot of efficiency vs. block length, L should be made and L chosen to maximize efficiency given any other constraints that are applicable.

Usually, the block convolutions are done by the FFT, but they could be done by any efficient, finite length method. One could use "rectangular transforms" or "number-theoretic transforms". A generalization of this method is presented later in the notes.

1.2 Fast Convolution by Overlap-Save

An alternative approach to the Overlap-Add can be developed by starting with segmenting the output rather than the input. If one considers the calculation of a block of output, it is seen that not only the corresponding input block is needed, but part of the preceding input block also needed. Indeed, one can show that a length $M+L-1$ segment of the input is needed for each output block. So, one saves the last part of the preceding block and concatenates it with the current input block, then convolves that with $h(n)$ to calculate the current output

2 Block Processing, a Generalization of Overlap Methods

Convolution is intimately related to the DFT. It was shown in The DFT as Convolution or Filtering[1] that a prime length DFT could be converted to cyclic convolution. It has been long known [47] that convolution can be calculated by multiplying the DFTs of signals.

An important question is what is the fastest method for calculating digital convolution. There are several methods that each have some advantage. The earliest method for fast convolution was the use of sectioning

[1] "The DFT as Convolution or Filtering" <http://cnx.org/content/m16328/latest/>

with overlap-add or overlap-save and the FFT [47], [51], [16]. In most cases the convolution is of real data and, therefore, real-data FFTs should be used. That approach is still probably the fastest method for longer convolution on a general purpose computer or microprocessor. The shorter convolutions should simply be calculated directly.

3 Introduction

The partitioning of long or infinite strings of data into shorter sections or blocks has been used to allow application of the FFT to realize on-going or continuous convolution [57], [31]. This section develops the idea of block processing and shows that it is a generalization of the overlap-add and overlap-save methods [57], [27]. They further generalize the idea to a multidimensional formulation of convolution [3], [12]. Moving in the opposite direction, it is shown that, rather than partitioning a string of scalars into blocks and then into blocks of blocks, one can partition a scalar number into blocks of bits and then include the operation of multiplication in the signal processing formulation. This is called distributed arithmetic [11] and, since it describes operations at the bit level, is completely general. These notes try to present a coherent development of these ideas.

4 Block Signal Processing

In this section the usual convolution and recursion that implements FIR and IIR discrete-time filters are reformulated in terms of vectors and matrices. Because the same data is partitioned and grouped in a variety of ways, it is important to have a consistent notation in order to be clear. The n^{th} element of a data sequence is expressed $h(n)$ or, in some cases to simplify, h_n. A block or finite length column vector is denoted $\underline{h}_n$ with n indicating the n^{th} block or section of a longer vector. A matrix, square or rectangular, is indicated by an upper case letter such as H with a subscript if appropriate.

4.1 Block Convolution

The operation of a finite impulse response (FIR) filter is described by a finite convolution as

$$y(n) = \sum_{k=0}^{L-1} h(k)\ x(n-k) \tag{4}$$

where $x(n)$ is causal, $h(n)$ is causal and of length L, and the time index n goes from zero to infinity or some large value. With a change of index variables this becomes

$$y(n) = \sum h(n-k)\ x(k) \tag{5}$$

which can be expressed as a matrix operation by

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} h_0 & 0 & 0 & \cdots & 0 \\ h_1 & h_0 & 0 & & \\ h_2 & h_1 & h_0 & & \\ \vdots & & & & \vdots \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \end{bmatrix}. \tag{6}$$

The H matrix of impulse response values is partitioned into N by N square sub matrices and the X and Y vectors are partitioned into length-N blocks or sections. This is illustrated for $N = 3$ by

$$H_0 = \begin{bmatrix} h_0 & 0 & 0 \\ h_1 & h_0 & 0 \\ h_2 & h_1 & h_0 \end{bmatrix} \qquad H_1 = \begin{bmatrix} h_3 & h_2 & h_1 \\ h_4 & h_3 & h_2 \\ h_5 & h_4 & h_3 \end{bmatrix} \qquad \text{etc.} \tag{7}$$

$$\underline{x}_0 = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \qquad \underline{x}_1 = \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} \qquad \underline{y}_0 = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} \qquad \text{etc.} \tag{8}$$

Substituting these definitions into (6) gives

$$\begin{bmatrix} \underline{y}_0 \\ \underline{y}_1 \\ \underline{y}_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} H_0 & 0 & 0 & \cdots & 0 \\ H_1 & H_0 & 0 & & \\ H_2 & H_1 & H_0 & & \\ \vdots & & & & \vdots \end{bmatrix} \begin{bmatrix} \underline{x}_0 \\ \underline{x}_1 \\ \underline{x}_2 \\ \vdots \end{bmatrix} \tag{9}$$

The general expression for the n^{th} output block is

$$\underline{y}_n = \sum_{k=0}^{n} H_{n-k}\, \underline{x}_k \tag{10}$$

which is a vector or block convolution. Since the matrix-vector multiplication within the block convolution is itself a convolution, (10) is a sort of convolution of convolutions and the finite length matrix-vector multiplication can be carried out using the FFT or other fast convolution methods.

The equation for one output block can be written as the product

$$\underline{y}_2 = [H_2 H_1 H_0] \begin{bmatrix} \underline{x}_0 \\ \underline{x}_1 \\ \underline{x}_2 \end{bmatrix} \tag{11}$$

and the effects of one input block can be written

$$\begin{bmatrix} H_0 \\ H_1 \\ H_2 \end{bmatrix} \underline{x}_1 = \begin{bmatrix} \underline{y}_0 \\ \underline{y}_1 \\ \underline{y}_2 \end{bmatrix}. \tag{12}$$

These are generalize statements of overlap save and overlap add [57], [27]. The block length can be longer, shorter, or equal to the filter length.

4.2 Block Recursion

Although less well-known, IIR filters can also be implemented with block processing [26], [17], [59], [9], [10]. The block form of an IIR filter is developed in much the same way as for the block convolution implementation of the FIR filter. The general constant coefficient difference equation which describes an IIR filter with recursive coefficients a_l, convolution coefficients b_k, input signal $x(n)$, and output signal $y(n)$ is given by

$$y(n) = \sum_{l=1}^{N-1} a_l\, y_{n-l} + \sum_{k=0}^{M-1} b_k\, x_{n-k} \tag{13}$$

using both functional notation and subscripts, depending on which is easier and clearer. The impulse response $h(n)$ is

$$h(n) = \sum_{l=1}^{N-1} a_l \, h(n-l) + \sum_{k=0}^{M-1} b_k \, \delta(n-k) \tag{14}$$

which can be written in matrix operator form

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ a_1 & 1 & 0 & & \\ a_2 & a_1 & 1 & & \\ a_3 & a_2 & a_1 & & \\ 0 & a_3 & a_2 & & \\ \vdots & & & & \vdots \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ \vdots \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ 0 \\ \vdots \end{bmatrix} \tag{15}$$

In terms of N by N submatrices and length-N blocks, this becomes

$$\begin{bmatrix} A_0 & 0 & 0 & \cdots & 0 \\ A_1 & A_0 & 0 & & \\ 0 & A_1 & A_0 & & \\ \vdots & & & & \vdots \end{bmatrix} \begin{bmatrix} \underline{h}_0 \\ \underline{h}_1 \\ \underline{h}_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} \underline{b}_0 \\ \underline{b}_1 \\ 0 \\ \vdots \end{bmatrix} \tag{16}$$

From this formulation, a block recursive equation can be written that will generate the impulse response block by block.

$$A_0 \, \underline{h}_n + A_1 \, \underline{h}_{n-1} = 0 \text{ for } n \geq 2 \tag{17}$$

$$\underline{h}_n = -A_0^{-1} A_1 \, \underline{h}_{n-1} = K \, \underline{h}_{n-1} \text{ for } n \geq 2 \tag{18}$$

with initial conditions given by

$$\underline{h}_1 = -A_0^{-1} A_1 A_0^{-1} \, \underline{b}_0 + A_0^{-1} \, \underline{b}_1 \tag{19}$$

This can also be written to generate the square partitions of the impulse response matrix by

$$H_n = K H_{n-1} \text{ for } n \geq 2 \tag{20}$$

with initial conditions given by

$$H_1 = K A_0^{-1} \, B_0 + A_0^{-1} \, B_1 \tag{21}$$

ane $K = -A_0^{-1} A_1$. This recursively generates square submatrices of H similar to those defined in (7) and (9) and shows the basic structure of the dynamic system.

Next, we develop the recursive formulation for a general input as described by the scalar difference equation (14) and in matrix operator form by

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ a_1 & 1 & 0 & & \\ a_2 & a_1 & 1 & & \\ a_3 & a_2 & a_1 & & \\ 0 & a_3 & a_2 & & \\ \vdots & & & & \vdots \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \end{bmatrix} = \begin{bmatrix} b_0 & 0 & 0 & \cdots & 0 \\ b_1 & b_0 & 0 & & \\ b_2 & b_1 & b_0 & & \\ 0 & b_2 & b_1 & & \\ 0 & 0 & b_2 & & \\ \vdots & & & & \vdots \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \end{bmatrix} \tag{22}$$

which, after substituting the definitions of the sub matrices and assuming the block length is larger than the order of the numerator or denominator, becomes

$$\begin{bmatrix} A_0 & 0 & 0 & \cdots & 0 \\ A_1 & A_0 & 0 & & \\ 0 & A_1 & A_0 & & \\ \vdots & & & & \vdots \end{bmatrix} \begin{bmatrix} \underline{y}_0 \\ \underline{y}_1 \\ \underline{y}_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} B_0 & 0 & 0 & \cdots & 0 \\ B_1 & B_0 & 0 & & \\ 0 & B_1 & B_0 & & \\ \vdots & & & & \vdots \end{bmatrix} \begin{bmatrix} \underline{x}_0 \\ \underline{x}_1 \\ \underline{x}_2 \\ \vdots \end{bmatrix}. \tag{23}$$

From the partitioned rows of (24), one can write the block recursive relation

$$A_0\,\underline{y}_{n+1} + A_1\,\underline{y}_n = B_0\,\underline{x}_{n+1} + B_1\,\underline{x}_n \tag{24}$$

Solving for $\underline{y}_{n+1}$ gives

$$\underline{y}_{n+1} = -A_0^{-1}A_1\,\underline{y}_n + A_0^{-1}B_0\,\underline{x}_{n+1} + A_0^{-1}B_1\,\underline{x}_n \tag{25}$$

$$\underline{y}_{n+1} = K\,\underline{y}_n + H_0\,\underline{x}_{n+1} + \tilde{H}_1\,\underline{x}_n \tag{26}$$

which is a first order vector difference equation [9], [10]. This is the fundamental block recursive algorithm that implements the original scalar difference equation in (14). It has several important characteristics.

- The block recursive formulation is similar to a state variable equation but the states are blocks or sections of the output [10], [34], [63], [64].
- The eigenvalues of K are the poles of the original scalar problem raised to the N power plus others that are zero. The longer the block length, the "more stable" the filter is, i.e. the further the poles are from the unit circle [9], [10], [63], [6], [7].
- If the block length were shorter than the denominator, the vector difference equation would be higher than first order. There would be a non zero A_2. If the block length were shorter than the numerator, there would be a non zero B_2 and a higher order block convolution operation. If the block length were one, the order of the vector equation would be the same as the scalar equation. They would be the same equation.
- The actual arithmetic that goes into the calculation of the output is partly recursive and partly convolution. The longer the block, the more the output is calculated by convolution and, the more arithmetic is required.
- It is possible to remove the zero eigenvalues in K by making K rectangular or square and N by N This results in a form even more similar to a state variable formulation [38], [10]. This is briefly discussed below in section 2.3.

- There are several ways of using the FFT in the calculation of the various matrix products in (25) and in (27) and (28). Each has some arithmetic advantage for various forms and orders of the original equation. It is also possible to implement some of the operations using rectangular transforms, number theoretic transforms, distributed arithmetic, or other efficient convolution algorithms [10], [63], [14], [13], [62], [49].
- By choosing the block length equal to the period, a periodically time varying filter can be made block time invariant. In other words, all the time varying characteristics are moved to the finite matrix multiplies which leave the time invariant properties at the block level. This allows use of z-transform and other time-invariant methods to be used for stability analysis and frequency response analysis [39], [40]. It also turns out to be related to filter banks and multi-rate filters [36], [35], [20].

4.3 Block State Formulation

It is possible to reduce the size of the matrix operators in the block recursive description (26) to give a form even more like a state variable equation [38], [10], [64]. If K in (26) has several zero eigenvalues, it should be possible to reduce the size of K until it has full rank. That was done in [10] and the result is

$$\underline{z}_n = K_1 \, \underline{z}_{n-1} + K_2 \, \underline{x}_n \tag{27}$$

$$\underline{y}_n = H_1 \, \underline{z}_{n-1} + H_0 \, \underline{x}_n \tag{28}$$

where H_0 is the same N by N convolution matrix, N_1 is a rectangular L by N partition of the convolution matrix H, K_1 is a square N by N matrix of full rank, and K_2 is a rectangular N by L matrix.

This is now a minimal state equation whose input and output are blocks of the original input and output. Some of the matrix multiplications can be carried out using the FFT or other techniques.

4.4 Block Implementations of Digital Filters

The advantage of the block convolution and recursion implementations is a possible improvement in arithmetic efficiency by using the FFT or other fast convolution methods for some of the multiplications in (10) or (25)[41], [42]. There is the reduction of quantization effects due to an effective decrease in the magnitude of the eigenvalues and the possibility of easier parallel implementation for IIR filters. The disadvantages are a delay of at least one block length and an increased memory requirement.

These methods could also be used in the various filtering methods for evaluating the DFT. This the chirp z-transform, Rader's method, and Goertzel's algorithm.

4.5 Multidimensional Formulation

This process of partitioning the data vectors and the operator matrices can be continued by partitioning (10) and (24) and creating blocks of blocks to give a higher dimensional structure. One should use index mapping ideas rather than partitioned matrices for this approach [3], [12].

4.6 Periodically Time-Varying Discrete-Time Systems

Most time-varying systems are periodically time-varying and this allows special results to be obtained. If the block length is set equal to the period of the time variations, the resulting block equations are time invariant and all to the time varying characteristics are contained in the matrix multiplications. This allows some of the tools of time invariant systems to be used on periodically time-varying systems.

The PTV system is analyzed in [61], [20], [19], [39], the filter analysis and design problem, which includes the decimation–interpolation structure, is addressed in [22], [40], [36], and the bandwidth compression problem in [35]. These structures can take the form of filter banks [58].

4.7 Multirate Filters, Filter Banks, and Wavelets

Another area that is related to periodically time varying systems and to block processing is filter banks [58], [29]. Recently the area of perfect reconstruction filter banks has been further developed and shown to be closely related to wavelet based signal analysis [20], [21], [28], [58]. The filter bank structure has several forms with the polyphase and lattice being particularly interesting.

An idea that has some elements of multirate filters, perfect reconstruction, and distributed arithmetic is given in [25], [23], [24]. Parks has noted that design of multirate filters has some elements in common with complex approximation and of 2-D filter design [55], [56] and is looking at using Tang's method for these designs.

4.8 Distributed Arithmetic

Rather than grouping the individual scalar data values in a discrete-time signal into blocks, the scalar values can be partitioned into groups of bits. Because multiplication of integers, multiplication of polynomials, and discrete-time convolution are the same operations, the bit-level description of multiplication can be mixed with the convolution of the signal processing. The resulting structure is called distributed arithmetic [11], [60]. It can be used to create an efficient table look-up scheme to implement an FIR or IIR filter using no multiplications by fetching previously calculated partial products which are stored in a table. Distributed arithmetic, block processing, and multi-dimensional formulations can be combined into an integrated powerful description to implement digital filters and processors. There may be a new form of distributed arithmetic using the ideas in [23], [24].

5 Direct Fast Convolution and Rectangular Transforms

A relatively new approach uses index mapping directly to convert a one dimensional convolution into a multidimensional convolution [12], [5]. This can be done by either a type-1 or type-2 map. The short convolutions along each dimension are then done by Winograd's optimal algorithms. Unlike for the case of the DFT, there is no savings of arithmetic from the index mapping alone. All the savings comes from efficient short algorithms. In the case of index mapping with convolution, the multiplications must be nested together in the center of the algorithm in the same way as for the WFTA. There is no equivalent to the PFA structure for convolution. The multidimensional convolution can not be calculated by row and column convolutions as the DFT was by row and column DFTs.

It would first seem that applying the index mapping and optimal short algorithms directly to convolution would be more efficient than using DFTs and converting them to convolution to be calculated by the same optimal algorithms. In practical algorithms, however, the DFT method seems to be more efficient [49].

A method that is attractive for special purpose hardware uses distributed arithmetic [11]. This approach uses a table look up of precomputed partial products to produce a system that does convolution without requiring multiplications [18].

Another method that requires special hardware uses number theoretic transforms [8], [37], [46] to calculate convolution. These transforms are defined over finite fields or rings with arithmetic performed modulo special numbers. These transforms have rather limited flexibility, but when they can be used, they are very efficient.

6 Number Theoretic Transforms for Convolution

6.1 Results from Number Theory

A basic review of the number theory useful for signal processing algorithms will be given here with specific emphasis on the congruence theory for number theoretic transforms [48], [30], [45], [37], [54].

6.2 Number Theoretic Transforms

Here we look at the conditions placed on a general linear transform in order for it to support cyclic convolution. The form of a linear transformation of a length-N sequence of number is given by

$$X(k) = \sum_{n=0}^{N-1} t(n,k)\ x(n) \tag{29}$$

for $k = 0, 1, \cdots, (N-1)$. The definition of cyclic convolution of two sequences is given by

$$y(n) = \sum_{m=0}^{N-1} x(m)\ h(n-m) \tag{30}$$

for $n = 0, 1, \cdots, (N-1)$ and all indices evaluated modulo N. We would like to find the properties of the transformation such that it will support the cyclic convolution. This means that if $X(k)$, $H(k)$, and $Y(k)$ are the transforms of $x(n)$, $h(n)$, and $y(n)$ respectively,

$$Y(k) = X(k)\ H(k). \tag{31}$$

The conditions are derived by taking the transform defined in (4) of both sides of equation (5) which gives

$$Y(k) = \sum_{n=0}^{N-1} t(n,k) \sum_{m=0}^{N-1} x(m)\ h(n-m) \tag{32}$$

$$= \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x(m)\ h(n-m)\ t(n,k). \tag{33}$$

Making the change of index variables, $l = n - m$, gives

$$= \sum_{m=0}^{N-1} \sum_{l=0}^{N-1} x(m)\ h(l)\ t(l+m,k). \tag{34}$$

But from (6), this must be

$$Y(k) = \sum_{n=0}^{N-1} x(n)\ t(n,k) \sum_{m=0}^{N-1} x(m)\ t(m,k) \tag{35}$$

$$= \sum_{m=0}^{N-1} \sum_{l=0}^{N-1} x(m)\ h(l)\ t(n,k)\ t(l,k). \tag{36}$$

This must be true for all $x(n)$, $h(n)$, and k, therefore from (9) and (11) we have

$$t(m+l,k) = t(m,k)\ t(l,k) \tag{37}$$

For $l = 0$ we have

$$t(m,k) = t(m,k)\ t(0,k) \tag{38}$$

and, therefore, $t(0,k) = 1$. For $l = m$ we have

$$t(2m,k) = t(m,k)\ t(m,k) = t^2(m,k) \tag{39}$$

For $l = pm$ we likewise have

$$t(pm, k) = t^p(m, k) \tag{40}$$

and, therefore,

$$t^N(m, k) = t(Nm, k) = t(0, k) = 1. \tag{41}$$

But

$$t(m, k) = t^m(1, k) = t^k(m, 1), \tag{42}$$

therefore,

$$t(m, k) = t^{mk}(1, 1). \tag{43}$$

Defining $t(1, 1) = \alpha$ gives the form for our general linear transform (4) as

$$X(k) = \sum_{n=0}^{N-1} \alpha^{nk} x(n) \tag{44}$$

where α is a root of order N , which means that N is the smallest integer such that $\alpha^N = 1$.

Theorem 1 The transform (13) supports cyclic convolution if and only if α is a root of order N and N^{-1} is defined.

This is discussed in [2], [4].

Theorem 2 The transform (13) supports cyclic convolution if and only if

$$N | O(M) \tag{45}$$

where

$$O(M) = gcd\{p_1 - 1, p_2 - 1, \cdots, p_l - 1\} \tag{46}$$

and

$$M = p_1^{r_1} p_2^{r_2} \cdots p_l^{r_l}. \tag{47}$$

This theorem is a more useful form of Theorem 1. Notice that $N_{max} = O(M)$.

One needs to find appropriate N, M, and α such that

- N should be appropriate for a fast algorithm and handle the desired sequence lengths.
- M should allow the desired dynamic range of the signals and should allow simple modular arithmetic.
- α should allow a simple multiplication for $\alpha^{nk} x(n)$.

We see that if M is even, it has a factor of 2 and, therefore, $O(M) = N_{max} = 1$ which implies M should be odd. If M is prime the $O(M) = M - 1$ which is as large as could be expected in a field of M integers. For $M = 2^k - 1$, let k be a composite $k = pq$ where p is prime. Then $2^p - 1$ divides $2^{pq} - 1$ and the maximum possible length of the transform will be governed by the length possible for $2^p - 1$. Therefore, only the prime k need be considered interesting. Numbers of this form are know as Mersenne numbers and have been used by Rader [52]. For Mersenne number transforms, it can be shown that transforms of length at least $2p$ exist and the corresponding $\alpha = -2$. Mersenne number transforms are not of as much interest because $2p$ is not highly composite and, therefore, we do not have FFT-type algorithms.

For $M = 2^k + 1$ and k odd, 3 divides $2^k + 1$ and the maximum possible transform length is 2. Thus we consider only even k. Let $k = s2^t$, where s is an odd integer. Then 2^{2^t} divides $2^{s2^t} + 1$ and the length of

the possible transform will be governed by the length possible for $2^{2^t}+1$. Therefore, integers of the form $M=2^{2^t}+1$ are of interest. These numbers are known as Fermat numbers [52]. Fermat numbers are prime for $0 \leq t \leq 4$ and are composite for all $t \geq 5$.

Since Fermat numbers up to F_4 are prime, $O(F_t)=2^b$ where $b=2^t$ and we can have a Fermat number transform for any length $N=2^m$ where $m \leq b$. For these Fermat primes the integer $\alpha=3$ is of order $N=2^b$ allowing the largest possible transform length. The integer $\alpha=2$ is of order $N=2b=2^{t+1}$. This is particularly attractive since α to a power is multiplied times the data values in (4).

The following table gives possible parameters for various Fermat number moduli.

t	b	$M=F_t$	N_2	$N_{\sqrt{2}}$	N_{max}	α for N_{max}
3	8	2^8+1	16	32	256	3
4	16	$2^{16}+1$	32	64	65536	3
5	32	$2^{32}+1$	64	128	128	$\sqrt{2}$
6	64	$2^{64}+1$	128	256	256	$\sqrt{2}$

Table 1

This table gives values of N for the two most important values of α which are 2 and $\sqrt{2}$. The second column give the approximate number of bits in the number representation. The third column gives the Fermat number modulus, the fourth is the maximum convolution length for $\alpha=2$, the fifth is the maximum length for $\alpha=\sqrt{2}$, the sixth is the maximum length for any α, and the seventh is the α for that maximum length. Remember that the first two rows have a Fermat number modulus which is prime and second two rows have a composite Fermat number as modulus. Note the differences.

The books, articles, and presentations that discuss NTT and related topics are [32], [37], [46], [8], [43], [44], [50], [53], [52], [1], [15], [2], [4]. A recent book discusses NT in a signal processing context [33].

References

[1] R. C. Agarwal and C. S. Burrus. Fast digital convolution using fermat transforms. In *Proceedings of the IEEE 25th Annual Southwestern Conference*, page 5388211;543, Houston, April 1973.

[2] R. C. Agarwal and C. S. Burrus. Fast convolution using fermat number transforms with applications to digital filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-22(2):87–97, April 1974. Reprinted in Number Theory in DSP, by McClellan and Rader, Prentice-Hall, 1979.

[3] R. C. Agarwal and C. S. Burrus. Fast one-dimensional digital convolution by multi-dimensional techniques. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-22(1):18211;10, February 1974. also in IEEE Press DSP Reprints II, 1979; and Number Theory in DSP, by McClellan and Rader, Prentice-Hall, 1979.

[4] R. C. Agarwal and C. S. Burrus. Number theoretic transforms to implement fast digital convolution. *Proceedings of the IEEE*, 63(4):5508211;560, April 1975. also in IEEE Press DSP Reprints II, 1979.

[5] R. C. Agarwal and J. W. Cooley. New algorithms for digital convolution. *IEEE Trans. on ASSP*, 25(2):3928211;410, October 1977.

[6] C. W. Barnes. Roundoff8211;noise and overflow in normal digital filters. *IEEE Transactions on Circuit and Systems*, CAS-26:1548211;155, March 1979.

[7] C. W. Barnes and S. Shinnaka. Block shift invariance and block implementation of discrete-time filters. *IEEE Transactions on Circuit and Systems*, CAS-27(4):6678211;672, August 1980.

[8] Richard E. Blahut. *Fast Algorithms for Digital Signal Processing.* Addison-Wesley, Reading, Mass., 1985.

[9] C. S. Burrus. Block implementation of digital filters. *IEEE Transactions on Circuit Theory*, CT-18(6):6978211;701, November 1971.

[10] C. S. Burrus. Block realization of digital filters. *IEEE Transactions on Audio and Electroacoustics*, AU-20(4):2308211;235, October 1972.

[11] C. S. Burrus. Digital filter structures described by distributed arithmetic. *IEEE Transactions on Circuit and Systems*, CAS-24(12):6748211;680, December 1977.

[12] C. S. Burrus. Index mapping for multidimensional formulation of the dft and convolution. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-25(3):2398211;242, June 1977.

[13] C. S. Burrus. Recursive digital filter structures using new high speed convolution algorithms. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, page 3638211;365, Hartford, CT, May 1977.

[14] C. S. Burrus and R. C. Agarwal. Efficient implementation of recursive digital filters. In *Proceedings of the Seventh Asilomar Conference on Circuits and Systems*, page 2808211;284, Pacific Grove, CA, November, 5 1973.

[15] C. S. Burrus and R. C. Agarwal. The use of number theoretic transforms for convolution. In *Presented at the IEEE Arden House Workshop on Digital Signal Processing*, Harriman, NY, January 1974.

[16] C. S. Burrus and T. W. Parks. *DFT/FFT and Convolution Algorithms.* John Wiley & Sons, New York, 1985.

[17] D. Chanoux. Synthesis of recursive digital filters using the fft. *IEEE Transactions on Audio and Electroacoustics*, AU-18:2118211;212, June 1970.

[18] Shuni Chu and C. S. Burrus. A prime factor fft algorithm using distributed arithmetic. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 30(2):2178211;227, April 1982.

[19] T. A. C. M. Claasen and W. F. G. Mecklenbraker. On stationary linear time8211;varying systems. *IEEE Trans. on Circuits and Systems*, 29(3):1698211;184, March 1982.

[20] R. E. Crochiere and L. R. Rabiner. *Multirate Digital Signal Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[21] Ingrid Daubechies. *Ten Lectures on Wavelets.* SIAM, Philadelphia, PA, 1992. Notes from the 1990 CBMS-NSF Conference on Wavelets and Applications at Lowell, MA.

[22] P. A. Franasek and B. Liu. On a class of time8211;varying filters. *IEEE Trans. on Information Theory*, 13:477, 1967.

[23] S. P. Ghanekar, S. Tantaratana, and L. E. Franks. High-precision multiplier8211;free fir filter realization with periodically time8211;varying coefficients. In *Paper Summaries for the 1992 DSP Workshop*, page 3.3.1, Starved Rock Lodge, Utica, Ill., 1992.

[24] S. P. Ghanekar, S. Tantaratana, and L. E. Franks. A class of high-precision multiplier8211;free fir filter realizations with periodically time8211;varying coefficients. *IEEE Transactions on Signal Processing*, 43(4):8228211;830, 1995.

[25] Sachin Ghanekar, Sawasd Tantaratana, and Lewis E. Franks. Implementation of recursive filters using highly quantized periodically time8211;varying coefficients. In *Proceedings of the ICASSP-91*, page 16258211;1628, Toronto, Canada, May 1991.

[26] B. Gold and K. L. Jordan. A note on digital filter synthesis. *Proceedings of the IEEE*, 56:17178211;1718, October 1968.

[27] B. Gold and C. M. Rader. *Digital Processing of Signals*. McGraw-Hill, New York, 1969.

[28] R. A. Gopinath and C. S. Burrus. Wavelet transforms and filter banks. In Charles K. Chui, editor, *Wavelets: A Tutorial in Theory and Applications*, page 6038211;655. Academic Press, San Diego, CA, 1992. Volume 2 in the series: Wavelet Analysis and its Applications.

[29] Ramesh A. Gopinath. *Wavelets and Filter Banks 8211; New Results and Applications*. Ph. d. thesis, Rice University, Houston, Tx, August 1992.

[30] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford, London, fourth edition, 1938, 1960.

[31] H. D. Helms. Fast fourier transform method of computing difference equations and simulating filters. *IEEE Trans. on Audio and Electroacoustics*, AU-15:858211;90, June 1967.

[32] Donald E. Knuth. *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms*. Addison-Wesley, Reading, MA, third edition, 1997.

[33] H. Krishna, B. Krishna, K.-Y. Lin, and J.-D. Sun. *Computational Number Theory and Digital Signal Processing*. CRC Press, Boca Raton, FL, 1994.

[34] C. M. Loeffler and C. S. Burrus. Equivalence of block filter representations. In *Proceedings of the 1981 IEEE International Symposium on Circuits and Systems*, pages 546–550, Chicago, IL, April 1981.

[35] C. M. Loeffler and C. S. Burrus. Periodically time8211;varying bandwidth compressor. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, page 6638211;665, Rome, Italy, May 1982.

[36] C. M. Loeffler and C. S. Burrus. Optimal design of periodically time varying and multirate digital filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-32(5):991–924, October 1984.

[37] J. H. McClellan and C. M. Rader. *Number Theory in Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1979.

[38] J. W. Meek and A. S. Veletsos. Fast convolution for recursive digital filters. *IEEE Transactions on Audio and Electroacoustics*, AU-20:938211;94, March 1972.

[39] R. A. Meyer and C. S. Burrus. A unified analysis of multirate and periodically time varying digital filters. *IEEE Transactions on Circuits and Systems*, CAS-22(3):1628211;168, March 1975.

[40] R. A. Meyer and C. S. Burrus. Design and implementation of multirate digital filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-24(1):538211;58, February 1976.

[41] S. K. Mitra and R. Gransekaran. A note on block implementation of iir digital filters. *IEEE Transactions on Circuit and Systems*, CAS-24(7), July 1977.

[42] S. K. Mitra and R. Gransekaran. Block implementation of recursive digital filters 8211; new structures and properties. *IEEE Transactions on Circuit and Systems*, CAS-25(4):2008211;207, April 1978.

[43] Douglas G. Myers. *Digital Signal Processing, Efficient Convolution and Fourier Transform Techniques.* Prentice-Hall, Sydney, Australia, 1990.

[44] P. J. Nicholson. Algebraic theory of finite fourier transforms. *Journal of Computer and System Sciences*, 5(2):5248211;547, February 1971.

[45] Ivan Niven and H. S. Zuckerman. *An Introduction to the Theory of Numbers.* John Wiley & Sons, New York, fourth edition, 1980.

[46] H. J. Nussbaumer. *Fast Fourier Transform and Convolution Algorithms.* Springer-Verlag, Heidelberg, Germany, second edition, 1981, 1982.

[47] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing.* Prentice-Hall, Englewood Cliffs, NJ, second edition, 1999. Earlier editions in 1975 and 1989.

[48] Oystein Ore. *Number Theory and Its History.* McGraw-Hill, New York, 1948.

[49] I. Pitas and C. S. Burrus. Time and error analysis of digital convolution by rectangular transforms. *Signal Processing*, 5(2):1538211;162, March 1983.

[50] J. M. Pollard. The fast fourier transform in a finite field. *Mathematics of Computation*, 25(114):3658211;374, April 1971.

[51] L. R. Rabiner and B. Gold. *Theory and Application of Digital Signal Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1975.

[52] Charles M. Rader. Discrete convolution via mersenne transforms. *IEEE Transactions on Computers*, 21(12):12698211;1273, December 1972.

[53] Charles M. Rader. Number theoretic convolution. In *IEEE Signal Processing Workshop*, Arden House, Harriman, NY, January 1972.

[54] Manfred R. Schroeder. *Number Theory in Science and Comminication.* Springer8211;Verlag, Berlin, second edition, 1984, 1986.

[55] R. G. Shenoy, Daniel Burnside, and T. W. Parks. Linear periodic systems and multirate filter design. Technical report GEO-002-92-16b, Schlumberger8211;Doll Research Note, September 1992.

[56] R. G. Shenoy, T. W. Parks, and Daniel Burnside. Fourier analysis of linear periodic systems and multirate filter design. In *Paper Summaries for the 1992 DSP Workshop*, page 2.4.1, Starved Rock Lodge, Utica, Ill., 1992.

[57] T. G. Stockham. High speed convolution and correlation. In *AFIPS Conf. Proc.*, volume 28, page 2298211;233. 1966 Spring Joint Computer Conference, 1966.

[58] P. P. Vaidyanathan. *Multirate Systems and Filter Banks.* Prentice-Hall, Englewood Cliffs, NJ, 1992.

[59] H. B. Voelcker and E. E. Hartquist. Digital filtering via block recursion. *IEEE Transactions on Audio and Electroacoustics*, AU-18:1698211;176, June 1970.

[60] S. A. White. Applications of distributed arithmetic to digital signal processing. *IEEE ASSP Magazine*, 6(3):48211;19, July 1989.

[61] L. A. Zadeh. Frequency analysis of variable networks. *Proceeding of the IRE*, 38(3):2918211;299, 1950.

[62] Y. Zalcstein. A note on fast convolution. *IEEE Transactions on Computers*, C-20:665, June 1971.

[63] Jan Zeman and Allen G. Lindgren. Fast digital filters with low round8211;off noise. *IEEE Transactions on Circuit and Systems*, CAS-28:7168211;723, July 1981.

[64] Jan Zeman and Allen G. Lindgren. Fast state8211;space decimator with very low round8211;off noise. *Signal Processing*, 3(4):3778211;388, October 1981.