Fast Convolution Algorithms

Overlap-add, Overlap-save

1 Introduction

One of the first applications of the (FFT) was to implement convolution faster than the usual direct method. Finite impulse response (FIR) digital filters and convolution are defined by

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

where, for an FIR filter, $x(n)$ is a length-N sequence of numbers considered to be the input signal, $h(n)$ is a length-L sequence of numbers considered to be the filter coefficients, and $y(n)$ is the filtered output. Examination of this equation shows that the output signal $y(n)$ must be a length-$(N + L - 1)$ sequence of numbers and the direct calculation of this output requires NL multiplications and approximately NL additions (actually, $(N-1)(L-1)$). If the signal and filter length are both length-N, we say the arithmetic complexity is of order N^2, $O(N^2)$. The goal is to calculate this convolution or filtering faster than directly implementing (1). The most common way to achieve "fast convolution" is to section or block the signal and use the FFT on these blocks to take advantage of the efficiency of the FFT. Clearly, one disadvantage of this technique is an inherent delay of one block length.

Indeed, this approach is so common as to be almost synonymous with fast convolution. The problem is to implement on-going, non-cyclic convolution with the finite-length, cyclic convolution that the FFT gives. An answer was quickly found in a clever organization of piecing together blocks of data using what is now called the *overlap-add* method and the *overlap-save* method. These two methods convolve length-L blocks using one length-L FFT, L complex multiplications, and one length-L inverse FFT.

2 Overlap-Add and Overlap-Save Methods for Fast Convolution

If one implements convolution by use of the FFT, then it is cyclic convolution that is obtained. In order to use the FFT, zeros are appended to the signal

or filter sequence until they are both the same length. If the FFT of the signal $x(n)$ is term-by-term multiplied by the FFT of the filter $h(n)$, the result is the FFT of the output $y(n)$. However, the length of $y(n)$ obtained by an inverse FFT is the same as the length of the input. Because the DFT or FFT is a periodic transform, the convolution implemented using this FFT approach is cyclic convolution which means the output of (1) is wrapped or aliased. The tail of $y(n)$ is added to it head — but that is usually not what is wanted for filtering or normal convolution and correlation. This aliasing, the effects of cyclic convolution, can be overcome by appending zeros to both $x(n)$ and $h(n)$ until their lengths are $N + L - 1$, and by then using the FFT. The part of the output that is aliased is zero and the result of the cyclic convolution is exactly the same as non-cyclic convolution. The cost is taking the FFT of lengthened sequences — sequences for which about half the numbers are zero. Now that we can do non-cyclic convolution with the FFT, how do we account for the effects of sectioning the input and output into blocks?

2.1 Overlap-Add

Because convolution is linear, the output of a long sequence can be calculated by simply summing the outputs of each block of the input. What is complicated is that the output blocks are longer than the input. This is dealt with by overlapping the tail of the output from the previous block with the beginning of the output from the present block. In other words, if the block length is N and it is greater than the filter length L, the output from the second block will overlap the tail of the output from the first block and they will simply be added. Hence the name: *overlap-add*. Figure 1 illustrates why the overlap-add method works, for $N = 10$, $L = 5$.

Combining the overlap-add organization with use of the FFT yields a very efficient algorithm for calculating convolution that is faster than direct calculation for lengths above 20 to 50. This cross-over point depends on the computer being used and the overhead needed by use of the FFTs.

2.2 Overlap-Save

A slightly different organization of the above approach is also often used for high-speed convolution. Rather than sectioning the input and then calculating the output from overlapped outputs from these individual input blocks, we will section the output and then use whatever part of the input contributes to that output block. In other words, to calculate the values

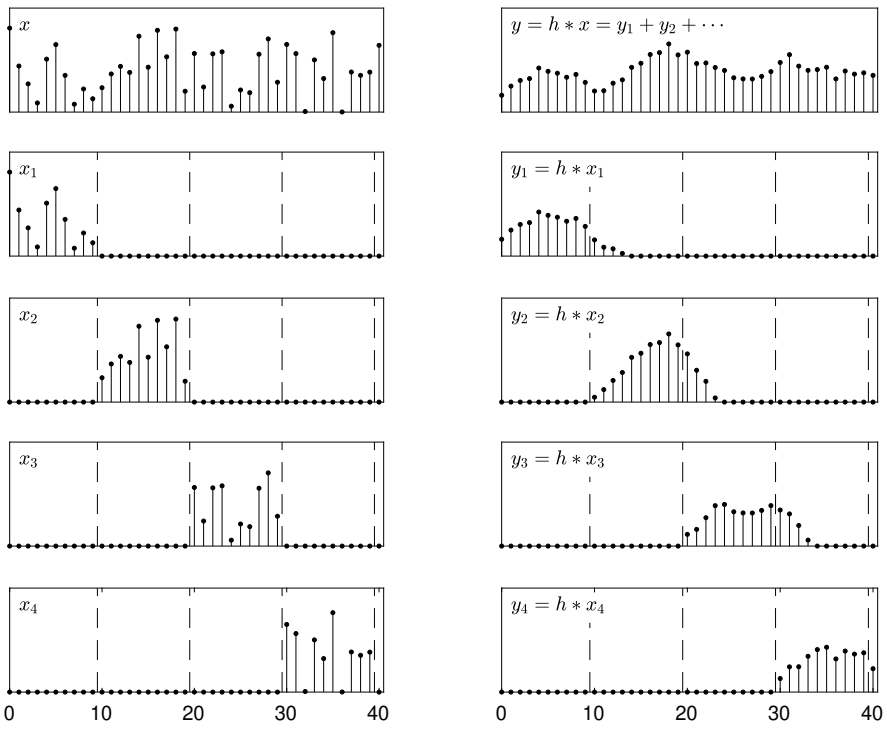

Figure 1: Overlap-Add Algorithm. The sequence $y(n)$ is the result of convolving $x(n)$ with an FIR filter $h(n)$ of length 5. In this example, $h(n) = 0.2$ for $n = 0, \ldots, 4$. The block length is 10, the overlap is 4. As illustrated in the figure, $x(n) = x_1(n) + x_2(n) + \cdots$ and $y(n) = y_1(n) + y_2(n) + \cdots$ where $y_i(n)$ is result of convolving $x_i(n)$ with the filter $h(n)$.

in a particular output block, a section of length $N + L - 1$ from the input will be needed. The strategy is to save the part of the first input block that contributes to the second output block and use it in that calculation. It turns out that exactly the same amount of arithmetic and storage are used by these two approaches. Because it is the input that is now overlapped and, therefore, must be saved, this second approach is called *overlap-save*.

This method has also been called *overlap-discard* because, rather than adding the overlapping output blocks, the overlapping portion of the output blocks are discarded. As illustrated in Figure 2, both the head and the tail of the output blocks are discarded. It may appear in Figure 2 that an FFT of length 18 is needed. However, with the use of the FFT (to get cyclic

convolution), the head and the tail overlap, so the FFT length is 14. (In practice, block lengths are generally chosen so that the FFT length $N+L-1$ is a power of 2).

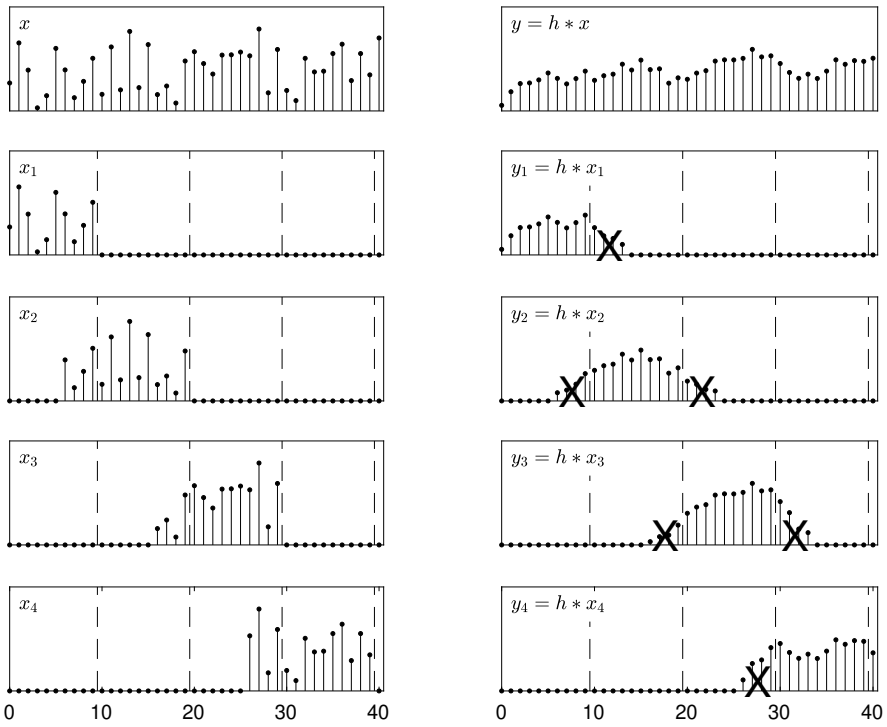

Figure 2: Overlap-Save Algorithm. The sequence $y(n)$ is the result of convolving $x(n)$ with an FIR filter $h(n)$ of length 5. In this example, $h(n) = 0.2$ for $n = 0, \ldots, 4$. The block length is 10, the overlap is 4. As illustrated in the figure, the sequence $y(n)$ is obtained, block by block, from the appropriate block of $y_i(n)$, where $y_i(n)$ is result of convolving $x_i(n)$ with the filter $h(n)$.

2.3 Use of the Overlap Methods

Because the efficiency of the FFT is $O(N \log(N))$, the efficiency of the overlap methods for convolution increases with length. To use the FFT for convolution will require one length-N forward FFT, N complex multiplications, and one length-N inverse FFT. The FFT of the filter is done once and stored rather done repeatedly for each block. For short lengths, direct con-

volution will be more efficient. The exact length of filter where the efficiency cross-over occurs depends on the computer and software being used.

If it is determined that the FFT is potentially faster than direct convolution, the next question is what block length to use. Here, there is a compromise between the improved efficiency of long FFTs and the fact you are processing a lot of appended zeros that contribute nothing to the output. An empirical plot of multiplication (and, perhaps, additions) per output point vs. block length will have a minimum that may be several times the filter length. This is an important parameter that should be optimized for each implementation. Remember that this increased block length may improve efficiency but it adds a delay and requires memory for storage.