Floating-point convolutions of length 3×2^k

Richard E. Crandall
Center for Advanced Computation, Reed College, Portland, OR

Abstract. Real signal floating-point cyclic convolutions of length $N = 3n = 3 \times 2^k$ can be performed with three real-signal FFT-based convolutions each of length n, or alternately with one real-signal and one complex-signal convolution, each of length n.

Mar 1997

1. Theory

Let $N = 3n = 3 \times 2^k$. One way to perform cyclic convolutions of such a length is to simply invoke a discrete Fourier transform (DFT) of length N, applying it three times in the following standard way: Let $x = \{x_0, x_1, \ldots, x_{N-1}\}$, with similar notation for y, and define the DFT of x as

$$\mathrm{DFT}(x)_k = \hat{x}_k = \sum_{j=0}^{N-1} x_j \exp(-w\pi ijk/N), \tag{1.1}$$

with $\hat{y}_k$ denoting in turn the DFT of y. The inverse DFT is then

$$x_j = \frac{1}{N} \sum_{k=0}^{N-1} \hat{x}_k \exp(+2\pi ijk/N). \tag{1.2}$$

As is standard, the cyclic convolution of length N, namely

$$(x \otimes y)_m = \sum_{j+k=m(\mathrm{mod}\ N)} x_j y_k, \tag{1.3}$$

can be obtained as

$$x \otimes y = \mathrm{DFT}^{-1}(\hat{x}\hat{y}), \tag{1.4}$$

where $\hat{x}\hat{y}$ is the dyadic, or element-wise product.

One drawback of this standard approach is that a length-$3n$ FFT may not be available, as most implementations of the FFT require input of length 2^k. We describe next an approach that uses only length-n FFTs. Not only are all signal lengths powers of two, but there are evident memory advantages, because the convolution problem is being broken up in to pieces.

An idea of J. P. Buhler is to start with the old notion (dating back to the seminal work of Winograd) that cyclic convolution is equivalent to polynomial multiplication, and use Chinese Remainder ideas for polynomials to reconstruct the final convolution. His new extension is to combine this approach with the disrete weighted transform (DWT) of Crandall and Fagin. Let the polynomial associated with signal x be defined as

$$x(t) = x_0 + x_1 t + x_2 t^2 + \ldots + x_{N-1} t^{N-1}. \tag{1.5}$$

Then the cyclic convolution can be written simply as

$$(x \otimes y)(t) = x(t)y(t)(\mathrm{mod}(t^N - 1)). \tag{1.6}$$

When $N = 3n$, and a non-trivial cube root of unity is available, we can factor $t^N - 1$, whence it is enough to know the three polynomial products

$$\begin{array}{l} x(t)y(t)(\mathrm{mod}(t^n - 1)) \\ x(t)y(t)(\mathrm{mod}(t^n - \omega)) \\ x(t)y(t)(\mathrm{mod}(t^n - \omega^2)). \end{array} \tag{1.7}$$

The first of these three parts can be calculated as a cyclic convolution of the reduced length n, while the other two can be obtained via two DWT's, each of length n. The DWT idea works a follows: In general, the expression

$$x(t)y(t)(\mathrm{mod}(t^n - A^n)) \tag{1.8}$$

can be obtained from the weighted convolution defined on signals x, y, now of reduced length n:

$$x \otimes_a y = \frac{1}{a}(ax \otimes ay), \tag{1.9}$$

where the weight signal a is given by

$$a = \{1, A, A^2, \ldots, A^{n-1}\}, \tag{1.10}$$

and signal products such as ax are dyadic products, with the inverse signal $1/a$ being the elementwise inverse.

These observations, together with Chinese Remainder reconstruction formulæ after the three length-n convolutions, are enough to establish an explicit algorithm.

2. The algorithm

In what follows, we use the assignment

$$A = \exp(-2\pi i/3n) \tag{2.1}$$

to generate the relevant weight signal. Note that $w = A^n$ is a non-trivial cube root of unity.

Algorithm

Input: Signals x, y, each of length $3n$.
Output: The cyclic convolution $x \otimes y$.

1) Create the weight signal $a = \{1, A, A^2, \ldots, A^{n-1}\}$.
2) Contract signal x into the length-n signal defined as

$$x'_j = x_j + x_{j+n} + x_{j+2n}, \qquad j = 0, 1, \ldots, n-1, \tag{2.2}$$

and contract y similarlly into y'.
3) Use real-signal length-n FFTs to compute the convolution

$$c = \mathrm{DFT}^{-1}(\widehat{x'}\widehat{y'}). \tag{2.3}$$

4) Contract signal x into the length-n signal defined as

$$x''_j = x_j + \omega x_{j+n} + \omega^2 x_{j+2n}, \qquad j = 0, 1, \ldots, n-1, \tag{2.4}$$

and contract y similarly into y''.

5) Multiply dyadically the weight signal a times x'' and y'', and use complex-signal FFTs and a final signal inversion by a to obtain the DWT:

$$d = \frac{1}{a}\mathrm{DFT}^{-1}(\widehat{ax''}\widehat{ay''}). \tag{2.5}$$

6) Construct the final cyclic convolution as the real-valued signal of length $3n$ formed as a union of the three length-n signals:

$$\begin{aligned} x \otimes y = \frac{1}{3}\{&\{c_j + 2\mathrm{Re}(d_j)\}_{j=0}^{n-1}, \\ &\{c_j - \mathrm{Re}(d_j) - \sqrt{3}\mathrm{Im}(d_j)\}_{j=0}^{n-1}, \\ &\{c_j - \mathrm{Re}(d_j) + \sqrt{3}\mathrm{Im}(d_j)\}_{j=0}^{n-1}\} \end{aligned} \tag{2.6}$$

This procedure is formally equivalent to the standard, length-$3n$ FFT approach. Here, however, all FFTs are of length-n, which reduces memory consumption. In addition, the algorithm may be modified in order to find the complex convolution d via real-valued FFTs, resulting in a slightly more complicated expression for the final union of Equation 2.6.

3. *Mathematica* implementation

A working *Mathematica* implementation of the length-$3n$ algorithm (only a dozen or so lines of source text) is available by e-mailing the author at "crandall@reed.edu".

Acknowledgments

The theory herein was worked out together with J. P. Buhler, who has developed methods for more general convolution lengths, and also for scenarios in which the relevant roots of unity do not exist in the algebraic domain of interest.