Integer convolution via split-radix fast Galois transform

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

Feb 1999

Abstract

Integer convolution can be effected, as is well known, via certain number-theoretical transforms. One particular transform, which we call a discrete Galois transform (DGT), can be used efficiently for either cyclic or negacyclic integer convolution. The DGT has the feature that, if an appropriate prime p for the field GF(p^2) be specified, the allowed power-of-two signal lengths can be quite large. Though the DGTs we consider involve complex arithmetic (amongst Gaussian integers $a + bi \mod p$), it turns out that the run lengths can, in certain settings, be halved. Thus, the fact of real-valued, integer input signals can be exploited along such lines, to enhance the performance of a resulting fast Galois transform (FGT) algorithm. Furthermore, split-radix FFT structure can be bestowed upon the FGT, again boosting efficiency for integer convolution.

1 Nomenclature

Consider the Galois field $GF(p^2)$, where $p = 2^q - 1$ is a Mersenne prime. Arithmetic in this field may proceed in the form of complex arithmetic involving Gaussian integers $a+bi$. The multiplicative group has order $p^2 - 1 = 2^q(2^q - 2) = 2^{q+1}(2^{q-1} - 1)$, so that there will be elements of any power of two up through 2^{q+1} inclusive. It is this existence of large power-of-two element orders that renders discrete transforms over GF(p^2) attractive. A theorem of [Creutzburg and Tasche 1989] gives a closed form for primitive roots. In fact, conveniently,

$$h = 2^{2^{q-2}} + (-3)^{2^{q-2}} i \mod p \tag{1}$$

is always a primitive root of order 2^{q+1}. It is a straightforward matter to generate directly from such a primitive root any required roots of order $2^k \le 2^{q+1}$, or, as will be required by our negacyclic variant, M-th roots of i.

We now define the discrete Galois transform (DGT). Let $x = \{x_0, \ldots x_{N-1}\}$ be a signal whose elements belong to GF(p^2). (These elements can be thought of as Gaussian integers $a + bi$, with each of a, b reduced mod p.) Then the transform in question is:

$$\hat{x}_k = \sum_{j=0}^{N-1} x_j g^{-jk} \mod p, \tag{2}$$

where g is a primitive root of order N in GF(p^2). The inverse transform is:

$$x_j = N^{-1} \sum_{k=0}^{N-1} \hat{x}_k g^{kj} \mod p. \tag{3}$$

These forward and inverse transforms are precisely analogous to the usual Fourier transforms, in that GF(p^2), for a Mersenne prime p (indeed for any prime $p = 3 \mod 4$), supports Gaussian integer arithmetic. Note that the specificiation of Mersenne primes is mainly to allow very large power-of-two run lengths. However, another advantage accrues: the (mod p) operation is especially efficient, involving only shifts and adds.

Now integer convolutions can be effected via these transforms. Heretofore we consider (real) integer input signals x, y; because we wish eventually to exploit the reality condition. For the cyclic convolution $x \otimes y$, where each of x, y is of length $N = 2^k$, we proceed:

Algorithm 1: cyclic convolution via DGT

1. Choose a Mersenne prime p such that every convolution element will be less than $p/2$ in magnitude;
2. Obtain the length-N DGTs $\hat{x}, \hat{y}$ of x,y respectively;
3. Multiply-mod dyadically to obtain a transform $\hat{z}$ according to:
$$\hat{z}_k = \hat{x}_k \hat{y}_k, \tag{4}$$
with all of the arithmetic performed $\mod p$.
4. The inverse DGT, z, of $\hat{z}$ is the cyclic convolution $z = x \otimes y \mod p$, which according to (1) is unambiguous $\mod p$ so z is in fact the desired cyclic convolution.

For other types of convolution, the steps are the same, except that a discrete weighted transform (DWT) over GF(p^2) should be employed. For example, negacyclic convolution can be effected simply, using a primitive N-th root of -1 in GF(p^2) [1, 2]. However, both cyclic and negacyclic cases can be enhanced dramatically, as we next investigate.

2 Exploiting reality of the input signals

Since the DGT with its Gaussian integer arithmetic is much like the standard complex DFT, we expect it possible to exploit the fact of real-valued input signals. Indeed, consider the "nested-complex" representation X of a real signal x:
$$X_j = x_{2j} + i x_{2j+1}, \tag{5}$$
where j now runs through $0, 1, \ldots, M-1$, with $M = N/2$. Now as is well known from the theory of standard FFTs, a shorter, length-M DGT on the (complex) signal X contains all the information one requires for inversion or convolution. The dyadic multiply-mod step is more complicated; we leave out the details and simply state the resulting dyadic relation below. Ther resulting overall scheme for convolution involves halved run lenghts at every step. We assume each of the (real) integer signals x, y to be of length $N = 2^k = 2M$:

Algorithm 2: Real-signal, cyclic convolution via DGT

1. Choose a Mersenne prime p such that every convolution element with be less than $p/2$ in magnitude;
2. Using the nested-complex representation (5), use a primitive M-th root of unity (call it G) to obtain two length-M DGTs $\hat{X}, \hat{Y}$ of the (complex) signals X, Y respectively (see Sections 4,5 on enhancements to the complex DGT itself);
3. Multiply-mod dyadically to obtain a transform $\hat{Z}$ according to:
$$\hat{Z}_k = (\hat{X}_k + \hat{X}^*_{-k})(\hat{Y}_k + \hat{Y}^*_{-k})$$
$$+2(\hat{X}_k\hat{Y}_k - \hat{X}^*_{-k}\hat{Y}^*_{-k}) - G^{-k}(\hat{X}_k - \hat{X}^*_{-k})(\hat{Y}_k - \hat{Y}^*_{-k}), \tag{6}$$
with all of the complex integer arithmetic performed modulo p.
4. Calculate the length-M inverse DGT, call it Z, of $\hat{Z}$. Then $Z/4$, though complex, is the nested-complex representation of the (real) cyclic convolution $z = x \otimes y \mod p$.

We see that cyclic convolution of two, length-N real integer signals can be effected thus with three, length-$N/2$ complex DGTs. One convenient choice of field for certain applications is to adopt $p = 2^{89} - 1$. Then integer convolutions of signals having 32-bit elements can be handled, for signal lengths up to about 2^{24}. In such cases the butterfly arithmetic within the DGT could be performed via 96-by-96 bit multiprecisions multiplies.

3 Negacyclic case

Again for real signals, there is a negacyclic convolution that enjoys all the benefits. This negacyclic variant is applicable to problems such as squaring modulo Fermat numbers. In fact, ongoing tests for the number:

$$F_{24} = 2^{2^{24}} + 1 \tag{7}$$

are using auto-negacyclic convolution (i.e. squaring modulo F_{24}) on residues having 2^{20} digits of 2^4 bits each. It turns out that the Mersenne prime $p = 2^{61} - 1$ works well for such a calculation. (The author has recently been informed that E. Mayer has completed a complete Pepin test implying F_{24} is composite. That test having used floating-point methods, the DGT approach underway has importance in pure-integer arithmetic verification of the character of F_{24}.)

It is convenient for the negacyclic case to define a different, let us say "folded" transform on a length-N real signal x:

$$X_j = x_j + i x_{j+N/2}. \tag{8}$$

It is not hard to see that now the full, negacyclic convolution can be obtained via a "right-angle" convolution of X sequences. That is, we twist the elements of X by powers of H, with H being an $N/2$-th root of i: That is to say, we can use the discrete weighted transform:

$$\hat{X}'_k = \sum_{j=0}^{N/2-1} X_j H^j G^{-jk}, \tag{9}$$

where G is an $N/2$-th root of unity, and do the same for a given signal $y \to Y$, to effect the desired negacyclic convolution of x, y:

Algorithm 3: Real-signal, negacyclic convolution via DGT

1. Choose a Mersenne prime p such that every convolution element with be less than $p/2$ in magnitude;

2. Using the folded-complex representation (8), twist elements $X'_k = X_k H^k$ (and same for Y_k) where H is an M-th root of i;

3. Use a primitive M-th root of unity (call it G) to obtain two length-M FGTs $\hat{X}', \hat{Y}'$ of the (complex) twisted signals X', Y' respectively (see Sections 4,5 on enhancements to the complex DGT itself);

4. Multiply-mod dyadically to obtain a transform $\hat{Z}'$ according to:

$$\hat{Z}'_k = \hat{X}'_k \hat{Y}'_k, \tag{10}$$

with all of the complex integer arithmetic performed modulo p.

5. Calculate the length-M inverse DGT, call it Z', of $\hat{Z}'$. Then the element collection $\{H^{-k} Z'_k\}$, though complex, is the folded-complex representation of the (real) negacyclic convolution of x, y mod p.

We note the simple dyadic multiply (step (4)) compared to the analgous step of Algorithm 2. Of course, Algorithm 3 involves twisting and untwisting of signals so the efficiency comparisons are nontrivial.

4 Fast mod and multiply operations

It is fortunate that a $\bmod p$ operation, when p is Mersenne, can be effected with shifts and adds only. Thus, within FGT butterflies most of the work is just (size p)-by-(size p) multiplies, with mods themselves consuming negligible time. Thus for example, when doing large-integer multiplication via the usual expedient of zero-padding of integer digit signals, the optimization of run length of DGTs will depend upon an assessment of the two quantities: a) the time to multiply two size-p integers $\bmod p$; and b) the number of arithmetic operations (essentially, the multiplies alone) in the DGT. Of course the operation count is $O(N \log N)$. However, the implied big-O constant can be reduced via split-radix butterfly format, as discussed next.

There is another, amusing enhancement that depends on properties of Mersenne numbers. Note that an inverse DGT of length 2^j has a prefactor $1/2^j$, which can also be thought of as 2^{q-j}. Either way, the peculiar property we have in mind is that multiplication by this prefactor is, modulo p, merely a circular shift, in one direction or the other.

As for multiplication of complex values in $GF(p^2)$, it should be noted that only 3 multiplies $\mod p$ (and some extra additions) are required to effect a full complex multiply; and this gain of 3/4 will be important when the chosen Mersenne prime p is large enough.

5 Split-radix FGT

Besides exploitation of reality of signal elements, another optimization is to use split-radix format for the butterflies of what we can call, as a fast means for calculating the DGT, the fast Galois transform (FGT). It is not necessarily true that any arbitrary transform has an FGT analog; for example there are real-signal floating point transforms (such as the Sorenson RVFFT) in which constants such as $\sqrt{2}$ arise; and the algebraic status of such constants is not always clear. But what is clear, upon inspection of the detailed arithmetic, is that the complex, split-radix FFT has an exact FGT analog. In fact, the important butterfly in the decimation-in-frequency, complex, split-radix format is of the general form

$$\begin{aligned}\{X_b, X_c, X_d, X_e\} := \{&X_b + X_d, X_c + X_e,\\ &X_b - X_d - ih^{-a}(X_c - X_e),\\ &X_b - X_d + ih^{-3a}(X_c - X_e)\}\end{aligned} \quad (11)$$

in which all multiplicative constants exist unambiguously in the field.

A symbolic realization of a split-radix FGT exists in the sotware released from Perfectly Scientific, Inc., Portland, OR, located at website:

`http://www.perfsci.com/`

References

[1] Crandall R E 1995, *Topics in Advanced Scientific Computation,* TELOS/Springer-Verlag.

[2] Crandall R E and Fagin B 1994, "Discrete weighted transforms and large-integer arithmetic," Math. Comp. 62, 205, 305–324.