EFFICIENT NTT ALGORITHMS FOR PRIME NUMBERS

Ryszard Stasiński
Instytut Elektroniki i Telekomunikacji
Politechnika Poznańska
Piotrowo 3A, PL-60-965 Poznań, POLAND

RESUME

Dans le conférence on présente la construction des algorithmes pour calculer
NTTs d'après l'idée de Rader. Pour N=p, où p est un nombre prime, non-Fermat,
les algorithmes besoins seulement O(p d_1 d_2 + ... + d_n) opérations, au contraire de
O(p d_1 d_2 ... d_n) pour le moyen direct. d_i sont les mutuellement primes diviseurs
de p-1. Ça signifie, que construits d'après cette idée algorithmes pour
transformation de Mersenne ne sont pas moins effectifs que ceux pour autres
NTTs. En général, d'après l'idée on peut construire des meilleures NTT modules,
utilisés dans les "FFT" algorithmes pour NTT. Le conférence contient aussi
quelques remarques concernant adoption pour NTT des algorithmes dérivés
originalemnt pour DFT.

SUMMARY

In the paper the construction of Rader's number theoretic transform algorithms
is described. It is shown that for N=p being non-Fermat prime numbers the
algorithms require significantly less operations than the known ones. Namely,
the number of operations is reduced from O(p d_1 d_2 ... d_n) to O(p d_1 + d_2 + ... + d_n),
where d_i are mutually prime divisors of p-1. If applied to Mersenne transforms
the approach results in algorithms which computational complexities are not
higher than those for other NTTs, e.g. pseudo-Fermat ones. In general, the
method can be used for improving small-N NTT modules in FFT-like algorithms.
The paper contains also some general remarks on the transformation of DFT
algorithms into those for number theoretic transforms.

1. INTRODUCTION

Number theoretic transforms (NTT) are used
for efficient computation of convolutions
using a special hardware. The most important
NTTs are Mersenne and Fermat transforms [1].
It is well known that the arithmetic for
Mersenne transform is especially simple.
Unfortunately, for the simplest realizations
its dimensions are equal to p, or 2p, p being
prime numbers. For other NTTs the problem
of efficient computation of transforms for sizes
being prime numbers is analogous to efficient
computation of small-N modules for the
discrete Fourier transform (DFT).

The idea of NTT can be treated as an
offspring of the idea of the DFT concept, so
it can be expected that some DFT algorithms
can be adapted to compute NTTs, too. Namely,
for N being prime numbers Rader's algorithms
are of interest [2], [3]. The same is true
for polynomial transforms (PTs), and, indeed,
it was shown that the use of Rader's PT
algorithms for N being prime numbers results
in dramatic reduction of the number of
operations [4]. Similarly as for PTs, the
computational complexity of a multiplication
in an NTT strongly depends on the form of a
multiplier (a shift vs a full ring
multiplication), so, the adaption of DFT
algorithms to the NTT case is linked with
some limitations. Note that similarities between NTTs and PTs are not accidental, as it was shown that some NTTs can be treated as PTs for digits [1].

In the paper the construction of Rader's NTT algorithms is described. The approach consists in transforming the problem of computing an N=2^p-point DFT into that of computing (p-1)p^{p-1}-point convolutions, p is an odd prime, s=r-1,r-2,...,0; [3]. It appears that if the convolutions are mapped into multidimensional ones on the basis of the rule from (5), the form of their coefficients remains the same as for NTTs, while the computation of the p-point convolution brackets goes into t stages, where t is a number of mutually prime divisors of p-1. Then, the number of operations decreases from \(O(p^3d_1d_2\cdots d_t)\) to \(O(p^2d_1d_2\cdots d_t)\), p-1=\(d_1d_2\cdots d_t\). The method gives very good results for N=2^p not being Fermat numbers.

2. ALGORITHMS

The most general definition of the DFT is the following [2]:

\[
X(k) = \sum_{n=0}^{N-1} x(n)W_N^{kn}, \quad k=0,1,\ldots,N-1. \tag{1}
\]

where \(X(k), x(n), W_N\) are elements of a commutative ring, and \(W_N\) is a (primitive) root of unity of order N. In the case of NTTs this is a ring of integers modulo \(M \equiv N\), or, sometimes, its extended version (6).

The most important feature of NTTs is that for some \(M \equiv W_N\) they are simply powers of 2:

\[
W_N = 2^{kn} \mod M \tag{2}
\]

Moreover, in the case of M being a Mersenne number (not necessarily prime) the computations are made simply in one's-complement arithmetic. Notice, however, that as:

\[
M = 2^p - 1, \quad p \text{ is prime}
\]

\(2^p \mod M = 1, \quad 2^r \mod M = 1 \text{ if } \gcd(r,p)\)

hence N in (1) N=2^p. Till now this fact was taken as an important limitation, as efficient NTT algorithms for N being prime numbers were not known. The use of more complicated arithmetic, and/or other M solve the problem only partially. Namely, in the case of NTTs the choice of N values is strongly restricted, and independent of the computational complexity criterion. [1], [8].

The Rader's DFT algorithm [2] exist for N being powers of prime numbers [3], [7]. It consists in an observation that:

\[
W_N = W_{Ckn} \mod N \tag{4}
\]

which means that calculations of the product \(W_N\) can be made in a ring modulo N. If N is a power of 2 and M is a prime number, the ring becomes an (extended) Galois field. We are interested in cases when N=2^p, and p is an odd prime.

* Some authors consider (1) as the definition of NTT. The DFT is then the NTT for complex numbers, see e.g. [6].

number. The Rader's algorithms for N=2^p exist [7], in the case of NTTs they are, however, neither effective, nor simple. For such N the elements of the field not being divisors of zero form a multiplicative group \(\mathbb{Z}_N\) which is cyclic, i.e.:

\[
a_1+\cdots+a_m, \quad 1+m \text{ is taken modulo } K \tag{5}
\]

where K is the rank of the group. The divisors of zero are simply multiples of p, hence:

\[
K = p^r - p^{r-1} \equiv p-1 \mod p^{r-1} \tag{6}
\]

So, the formula on the DFT (1) can be rewritten as follows:

\[
X(a_k) = \sum_{n=0}^{K-1} x(n)W_N^{n+k} \tag{7a}
\]

\[
X(pk') = \sum_{n=0}^{N-1} x(n)W_N^{kn}, \quad k'=0,1,\ldots,N/p-1; \tag{7b}
\]

where:

\[
X(a_k) = \sum_{n=0}^{N-1} x(n)W_N^{kn+p} \tag{7c}
\]

The summation in (7a) is equivalent to the K-point circular correlation, which can be computed using circular convolution algorithms. X(pk') and X(k) can be computed using N/p-point DFT algorithms [3].

Notice that p^{r-1} and p-1 numbers in (6) are mutually prime. It means that the convolution can be mapped into multidimensional d-1 dipoint convolution. p-1=\(d_1d_2\cdots d_t\), p-1=\(d_1d_2\cdots d_t\), t is a number of mutually prime divisors [5], and:

\[
p-1 = \prod_{i=1}^t d_i \tag{8}
\]

The p^{r-1}-point convolution is de facto a polynomial product modulo cyclotomic polynomial for Z^{r-1} [3], however, this fact need not be used here.

3. ARITHMETICAL COMPLEXITY OF ALGORITHM

Consider N=p. In this case (7):

\[
X(pk') = X(0) = \sum_{n=0}^{N-1} x(n) \tag{9a}
\]

\[
\hat{X}^k = x(0) \tag{9b}
\]

If we compute circular convolutions directly, the coefficients of convolutions are [1], [5]:

\[
W_N = 1, \quad m_1=0,1,\ldots,d_1-1; \quad D_1 = (p-1)/d_1 \tag{10}
\]

see (8), with \(W_N = 2\) (2). Any further transformation of algorithms causes that the coefficients become more complicated. This means that the computation of Rader's NTT algorithm requires 2kp^{r-1} additions (9) plus operations due to direct computation of t-dimensional d_1d_2\cdots d_t dipoint circular convolution. An d-point convolution can be
computed using \( d^2 \) shifts and \( (d-1) \) additions, so, the overall algorithm requires:

\[
S(KNp) = (p-1) \sum d_i \text{ shifts, and} \tag{11a}
\]

\[
A(KNp) = (p-1)2^t \sum (d_i -1) \text{ additions.} \tag{11b}
\]

In the case of direct method:

\[
\sum \frac{1}{N} \left( \sum K_i \right) \text{ shifts, and} \tag{12}
\]

which means that (1):

\[
X(N-1) = X0 - \sum \sum \frac{K_i}{N} \text{ shifts, and} \tag{13}
\]

the fact was used in Table I for improving FT algorithms. Taking into account (13) the direct method results in:

\[
S(Kp) = (p-1)(p-2) \text{ shifts, and} \tag{14a}
\]

\[
A(Kp) = (p-1) \text{ additions.} \tag{14b}
\]

Comparing (11) and (14) we can see that the Rader's NTT algorithm require asymptotically \( O(p^2d) \) operations, in contrast to \( O(p^2d) \) for direct method. Table I shows that indeed, except for \( N \) being Fermat prime numbers improvements due to Rader's NTT algorithm are dramatic. Reductions of numbers of operations are especially big when \( p-1 \) has many small divisors, e.g. for \( p=13 \), 31, 61, 71, 127, and 211. For Fermat prime numbers Rader's NTT algorithm is identical to the "ordinary" direct method, hence, the results are somewhat worse than those implied by (14).

The Rader's NTT algorithms for powers of a prime are not interesting here. Namely, they contain \( p-1 \) in fact \( p-2 \) \( \left( \frac{p-1}{2} \right) \) \( p-1 \)-point, and \( p-1 \)-point circular convolutions to be computed, \( N=p^t \). For the FFT-like algorithms the operations consist of \( p^t-1 \) \( p-1 \)-point operations. Of course, \( p \) has no divisors, so, even for \( p \approx 2 \) FFT-like algorithms are better than Rader's ones, see also [4].

4. SUMMARY AND CONCLUSION

In the paper the construction of number theoretic transform algorithms using the idea of Rader is described. It is observed that if \( N \) is an odd non-Fermat prime number the approach results in a class of algorithms having computational complexity of rank \( O(p^2d) \), where \( d_i \) are mutually prime divisors of \( p-1 \), in contrast to \( O(p^2d) \) for direct method. As it is shown in Table I, the new algorithms are really very efficient, especially for big numbers of divisors of \( p-1 \).

The introduction of Rader's NTT algorithms causes that the computational complexity of long Mersenne transforms reduces to the level characteristic of other NTTs. Consider, for example, the 64-point pseudo-Fermat transform [1]. For the structure of the prime factor algorithm \( (2\times2)\times17 \) it requires \( 4\times340 \) shifts due to \( 17 \)-point algorithms, Table I, plus \( 2\times137 \) ones for rotation factors, which gives \( 977 \) shifts, and \( 2\times34\times4\times272 \) additions, Table I. As can be seen, the algorithm is worse than the Mersenne ones for \( N=64 \), 67, 71, and only slightly better than that for \( N=73 \). Additionally, the pseudo-Fermat transform requires a special stage of final reductions, and has approximately two times smaller number of effective bits of results [1]. Of course, the idea may be used for improving non-Mersenne NTTs, too.

\[ \text{Table I } \]

<table>
<thead>
<tr>
<th>( p )</th>
<th>( p-1 )</th>
<th>( d_i )</th>
<th>Rader (11) shifts/adds</th>
<th>Direct (14) shifts/adds</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1/2</td>
<td>0/2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2/6</td>
<td>2/6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
<td>16/20</td>
<td>12/20</td>
</tr>
<tr>
<td>7</td>
<td>2 * 3</td>
<td>5</td>
<td>30/30</td>
<td>30/42</td>
</tr>
<tr>
<td>11</td>
<td>2 * 5</td>
<td>7</td>
<td>70/70</td>
<td>90/110</td>
</tr>
<tr>
<td>13</td>
<td>4 * 3</td>
<td>8</td>
<td>84/84</td>
<td>132/156</td>
</tr>
<tr>
<td>17</td>
<td>16</td>
<td>16</td>
<td>256/272</td>
<td>240/272</td>
</tr>
<tr>
<td>19</td>
<td>16</td>
<td>11</td>
<td>258/192</td>
<td>306/342</td>
</tr>
<tr>
<td>23</td>
<td>11</td>
<td>11</td>
<td>286/266</td>
<td>402/506</td>
</tr>
<tr>
<td>29</td>
<td>11</td>
<td>11</td>
<td>308/308</td>
<td>796/812</td>
</tr>
<tr>
<td>31</td>
<td>10</td>
<td>10</td>
<td>300/270</td>
<td>870/930</td>
</tr>
<tr>
<td>37</td>
<td>13</td>
<td>13</td>
<td>458/458</td>
<td>1280/1320</td>
</tr>
<tr>
<td>41</td>
<td>13</td>
<td>13</td>
<td>520/520</td>
<td>1560/1640</td>
</tr>
<tr>
<td>43</td>
<td>13</td>
<td>12</td>
<td>560/542</td>
<td>1722/1808</td>
</tr>
<tr>
<td>47</td>
<td>20</td>
<td>18</td>
<td>1150/1150</td>
<td>2070/2122</td>
</tr>
<tr>
<td>53</td>
<td>20</td>
<td>17</td>
<td>2652/2752</td>
<td>7232/7396</td>
</tr>
<tr>
<td>59</td>
<td>29</td>
<td>17</td>
<td>7308/7278</td>
<td>3906/3422</td>
</tr>
<tr>
<td>61</td>
<td>22</td>
<td>14</td>
<td>800/810</td>
<td>4800/4422</td>
</tr>
<tr>
<td>67</td>
<td>18</td>
<td>17</td>
<td>1256/1224</td>
<td>5112/5256</td>
</tr>
<tr>
<td>71</td>
<td>22</td>
<td>17</td>
<td>1404/1348</td>
<td>6068/6182</td>
</tr>
<tr>
<td>73</td>
<td>17</td>
<td>19</td>
<td>1692/1672</td>
<td>7066/7382</td>
</tr>
<tr>
<td>79</td>
<td>17</td>
<td>18</td>
<td>2228/2124</td>
<td>10300/10508</td>
</tr>
<tr>
<td>89</td>
<td>19</td>
<td>19</td>
<td>2512/2412</td>
<td>15780/16002</td>
</tr>
<tr>
<td>127</td>
<td>17</td>
<td>17</td>
<td>3705/3710</td>
<td>43950/44310</td>
</tr>
</tbody>
</table>
