Polynomial multiplication over prime fields : a comparison

H. Riffaud de Turckheim, M. de Castelbajac

We're aiming to compare the efficiency of a couple of well-known algorithms for polynomial multiplication over prime fields, such as Karatsuba and Toom-Cook algorithms versus the naive multiplication method. We restrict ourselves to prime fields – *i.e.* $\frac{\mathbb{Z}}{p\mathbb{Z}}$ – with a prime of size at most thirty bits.

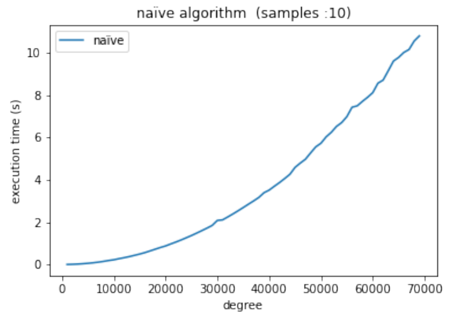
Contents

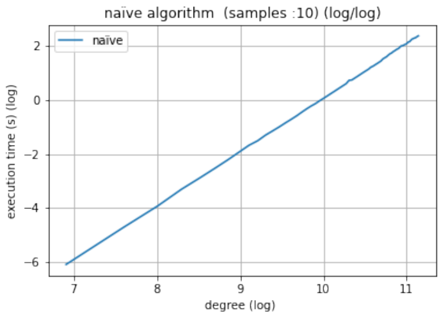
1. Naive Algorithm

1.1 presentation

The easiest and naïve way to multiply polynomials, also known as the "school book" multiplication. The method multiply each coefficient of one polynomial with all the coefficients of the other one. This method complexity is: $O(n^2)$.

1.2 complexity





As we can see on the right picture , the slope of the curve is ≈ 2.0 . This confirms the $O(n^2)$ complexity .

2. Karatsuba Algorithm

2.1 presentation

The Karatsuba algorithm for polynomial multiplication is a variation of the Karatsuba multiplication algorithm for large numbers [2]. This algorithm is based on a "divide and conquer" strategy. The complexity of this algorithm is O($n^{1.59}$).

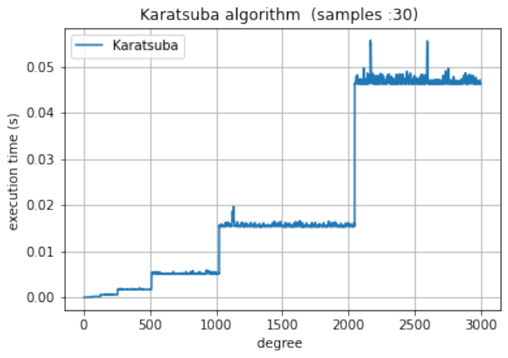
Algorithm 1 Karatsuba

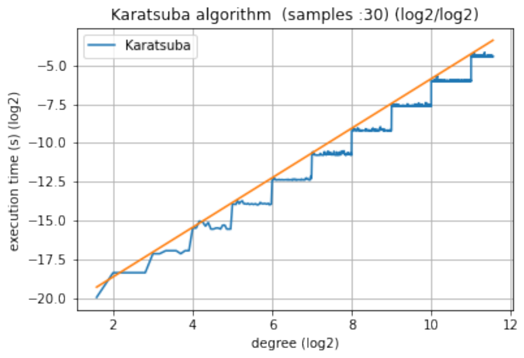
```
  Input : $P, Q$ two $2^n$ sized polynomials,n
  Output : $C = P \cdot Q$

if n$\leq$ 2 then
  return NaiveMultiplication($P$,$Q$)
end if
m $\leftarrow \lceil \frac{n}{2} \rceil$
P $\leftarrow P_0 \cdot x^m + P_1$
Q $\leftarrow Q_0 \cdot x^m + Q_1$
$R_0 \leftarrow$Karatsuba$(P_0, Q_0, \frac{n}{2})$
$R_1 \leftarrow$Karatsuba$(P_1, Q_1, \frac{n}{2})$
$T_0 \leftarrow P_0 + P_1$
$T_1 \leftarrow Q_0 + Q_1$
$R_2 \leftarrow R_2 - R_1 - R_0$

return $R_0 \cdot x^{2n} + R_2 \cdot x^n + R_1$
```

2.2 complexity





As we can see on the right picture , the complexity of this algorithm is a "stair case" one. This is due to the fact that we enlarge each polynomial to the next power of two. We traced a line to determine the complexity of our algorithm. Our algorithm has a $\approx$O($n^{1.594}$) complexity which is not far from the theoretical one.

3. Toom-Cook Algorithm(s)

3.1 presentation

The Toom-Cook$_k$ multiplication is a generalization of the Karatsuba multiplication where the instances are split in k parts instead of two.
Past the splitting step, we aim to solve the polynomial interpolation problem for each polynomial through a Vandermonde matrix by evaluating each polynomial in $k-1$ points, which results in solving an interpolation problem in $2k-2$ different points. We give below an example for two polynomials $A(x)$ and $B(x)$ of degree n [1]. We can write them as :

$$A(x) = \alpha_{k-1} * x^{(n/k)(k-1)} + \cdots + \alpha_0$$

$$B(x) = \beta_{k-1} * x^{(n/k)(k-1)} + \cdots + \beta_0$$

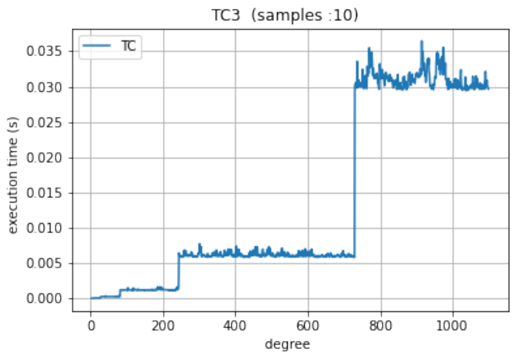
where $\alpha_{k-1} = a_{n-1} * x^{n/k-1} + \cdots + a_{n-n/k}$ and $\alpha_{k-2} = a_{n-n/k-1} * x^{n/k-1} + \cdots + a_{n-2n/k}$ and so on and similarly for β's.
Then since the two polynomials are properly split, we have the *evaluation* step:

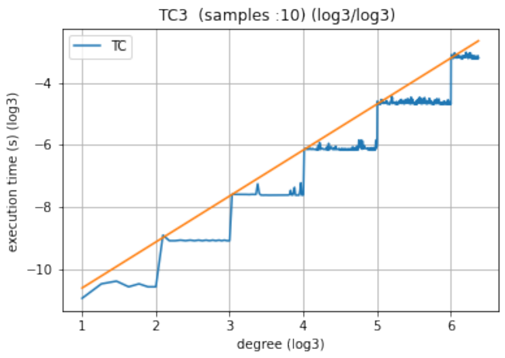
$$\begin{pmatrix} A(p_0) \\ A(p_1) \\ \vdots \\ A(p_{2k-2}) \end{pmatrix} = \begin{pmatrix} p_0^0 & p_0^1 & \dots & p_0^{k-1} \\ p_1^0 & p_1^1 & \dots & p_1^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ p_{2k-2}^0 & p_{2k-2}^1 & \dots & p_{2k-2}^{k-1} \end{pmatrix} = \begin{pmatrix} \alpha_0 \\ \alpha_1) \\ \vdots \\ \alpha_{k-1}) \end{pmatrix}$$

And we do similarly for B. The two obtained vectors are multiplied pointwise during the *multiplication* step. The final *interpolation* step leads to computing the resulting split polynomial by inverting the matrix shown above and solving the system using the combined vector obtained with the previous pointwise multiplication. The arithmetic complexity is bounded by $\mathcal{O}(n^{log_3^5})$ for Toom-Cook$_3$ or more generally by $\mathcal{O}(n^{log_k^{2k-1}})$.

3.2 complexity

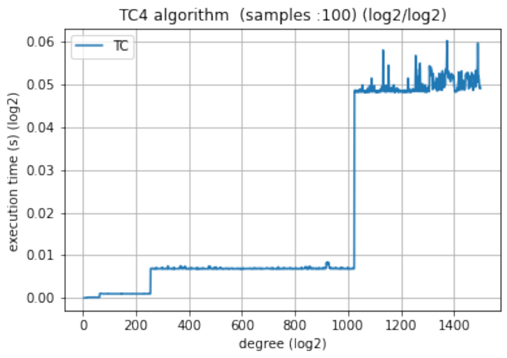
3.2.1 TC3

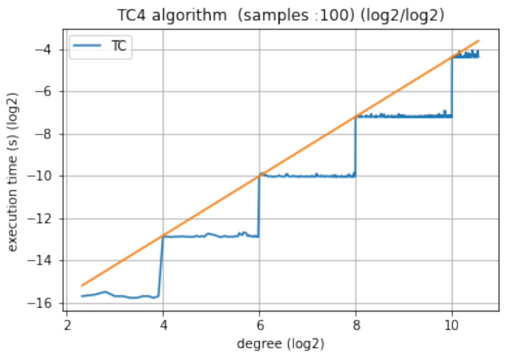




As we can see on the right picture , the complexity of this algorithm is a ”stair case” one. This is due to the fact that we enlarge each polynomial to the next power of three. Our algorithm has a $\approx$O($n^{1.48}$) complexity which is not far from the theoretical one (O($n^{1.464}$) for TC3).

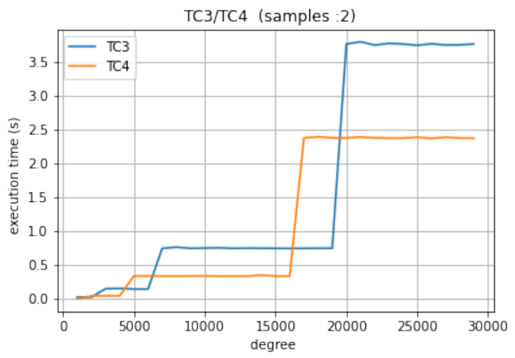
3.2.2 TC4

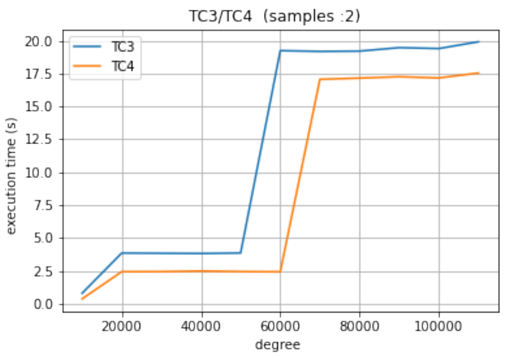




Our algorithm has a $\approx$O($n^{1.406}$) complexity which is not far from the theoretical one (O($n^{1.40}$) for TC4).

3.2.3 TC3 vs TC4





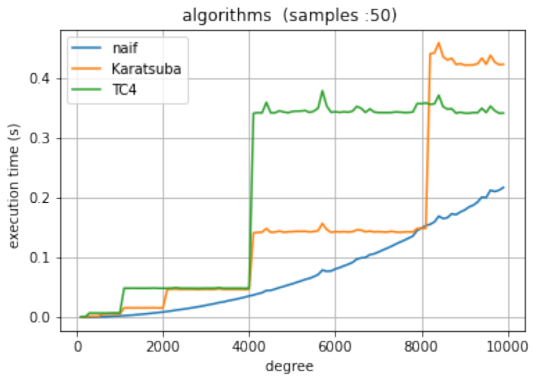
As we can see, TC4 is faster than TC3.

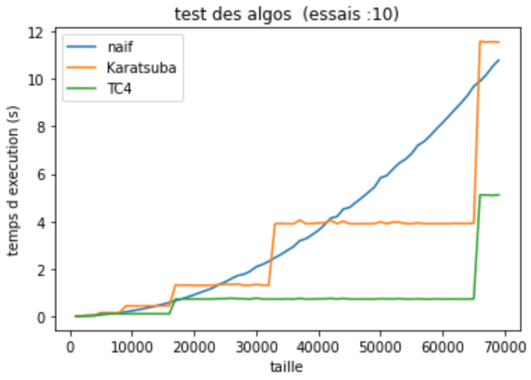
4. validity of our algorithms

To ensure the correctness of our algorithms we tried to execute them on multiple polynomials with various modulo, sizes and random coefficients.Then we compared the answers.
We can suppose that the naïve algorithm, Karatsuba, TC3, TC4. are valid.

5. Results

Now we can determine the optimal degree where it's better to use the naive algorithm, Karatsuba or TC4.





- **it's better to use Karatsuba over the naive algorithm for polynomials with a degree $\geq 2^{16}$.**
- **it's better to use TC for polynomials over the naive algorithm with a degree $\geq 2^{14}$.**

References

[1] Bermudo Mera, J. M., Karmakar, A., and Verbauwhede, I. Time-memory trade-off in Toom-Cook multiplication: an application to module-lattice based cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (Mar. 2020), 222–244.

[2] Weimerskirch, A., and Paar, C. Generalizations of the Karatsuba Algorithm for Efficient Implementations. *IACR Cryptol. ePrint Arch.* (2006), 224.