Multiplier des entiers en temps O (n log n)

Joris van Der Hoeven

▶ **To cite this version:**

Joris van Der Hoeven. Multiplier des entiers en temps O (n log n). Gazette du GDR IM, 2021. hal-03278647

HAL Id: hal-03278647
https://hal.science/hal-03278647v1

Submitted on 5 Jul 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Multiplier des entiers en temps $O(n \log n)$*†

par Joris van der Hoeven
CNRS, LIX (UMR 7161)
1, rue Honoré d'Estienne d'Orves
Bâtiment Alan Turing, CS35003
91120 Palaiseau, France

Mars 2021

Résumé

Il a été démontré récemment que deux entiers de n chiffres peuvent être multipliés en temps $O(n \log n)$ [12]. L'existence d'un tel algorithme fut conjecturé en 1971 par Schönhage et Strassen [23]. Ils émettaient également leurs doutes sur l'existence d'une méthode encore plus rapide, ce que l'on ignore toujours.

Dans cet article, je présenterai brièvement une longue histoire et je poursuivrai avec un aperçu du nouvel algorithme.

Introduction

Quelle est la meilleure façon de multiplier deux entiers ? Cette question, simple en apparence, est peut-être le plus ancien problème mathématique non résolu. Ainsi, ce problème est au moins dix fois plus ancien que la conjecture de Fermat, alias « théorème de Wiles ».

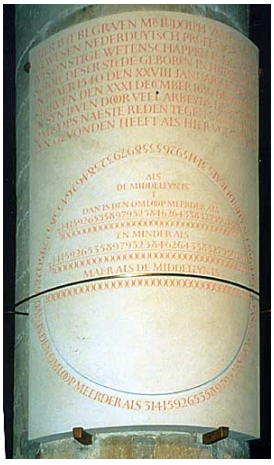

Figure 1. Pierre commémorative dans le Pieterskerk à Leyde, avec une reconstruction du texte original de la pierre tombale de Ludolph van Ceulen (1540–1610), qui a passé vingt-cinq ans de sa vie à calculer 35 décimales de π.

Il faut bien sûr préciser ce que nous entendons par « meilleur ». Pour cela, il a fallu attendre l'invention de la machine de Turing : équipée d'un modèle de calcul précis, on peut définir le nombre $M(n)$ d'étapes nécessaires pour multiplier deux nombres de n chiffres. La meilleure méthode est alors celle pour laquelle $M(n)$ augmente aussi lentement que possible en fonction de n.

Prenons par exemple la méthode de l'école primaire, qui consiste à multiplier chaque chiffre du premier nombre par chaque chiffre du deuxième nombre, en ajoutant les résultats de manière appropriée. Cela donne $M(n) = O(n^2)$: il existe une constante $C > 0$ avec $M(n) \leqslant C n^2$ pour tout n.

Afin de rendre notre question principale indépendante de la machine de Turing en cher et en os sur laquelle nous faisons notre calcul, nous nous intéressons uniquement au temps de calcul à un facteur constant près. Une méthode 100 fois plus rapide n'est donc pas une amélioration significative, mais un algorithme $\log\log\log n$ fois plus rapide est un énorme pas en avant.

C'est probablement Kolmogorov qui a été le premier à formuler notre problème ainsi, dans un séminaire célèbre à Moscou. Emporté par son enthousiasme, il avait également

Figure 2. Andrej Kolmogorov.

*. Cet article est une traduction libre de [15].
†. Ce document a été rédigé avec GNU TeXmacs [16].

conjecturé que $n^2 = O(\mathrm{M}(n))$. Car, raissona-t-il, s'il y avait une meilleure méthode que celle de l'école, on l'aurait bien trouvé depuis six mille ans.

La formulation précise d'un problème facilite la recherche de solutions. Mais l'absence d'une telle formulation n'exclut pas d'éventuels progrès. Les Babyloniens ont déjà calculé $\sqrt{2}$ jusqu'à seize chiffres derrière la virgule. Un jour, on trouvera peut-être une tablette d'argile avec des méthodes inédites de multiplication à grande vitesse. Les amateurs du nombre π, comme mon compatriote Ludolph van Ceulen, auraient pu en tirer profit (voir la figure 1).

Karatsuba

La conjecture de Kolmogorov ne fit pas long feu. Après trois semaines, il a été surpris par un jeune élève avec une méthode de multiplication en $O(n^{\log 3/\log 2})$ étapes. Kolmogorov rédigea immédiatement cette nouvelle trouvaille, la publia sous le nom de son créateur, Karatsuba, et la regroupa avec un autre article qui n'avait rien à voir [18, 17].

Expliquons l'idée de Karatsuba avec un exemple :

$$\begin{aligned} u &= 220629012020 \\ v &= 314159265358. \end{aligned}$$

Nous commençons par couper les deux nombres en deux :

$$\begin{aligned} u &= \overbrace{220629}^{a} \times \overbrace{1000000}^{x} + \overbrace{012020}^{b} \\ v &= \underbrace{314159}_{c} \times 1000000 + \underbrace{265358}_{d} \end{aligned}$$

Ensuite, nous faisons deux additions et trois multiplications :

$$\begin{aligned} a+b &= 232649 \\ c+d &= 579517 \\ ac &= 069312586011 \\ bd &= 003189603160 \\ (a+b)(c+d) &= 134824050533 \end{aligned}$$

Figure 3. Anatoly Karatsuba.

Maintenant, nous remarquons qu'il suffit de deux soustractions pour trouver

$$\begin{aligned} ad+bc &= (a+b)(c+d) - ac - bd \\ &= 62321861362. \end{aligned}$$

Enfin, on a

$$\begin{aligned} uv &= (ax+b)(cx+d) \\ &= acx^2 + (ad+bc)x + bd, \end{aligned}$$

et on termine avec une dernière addition :

$$\begin{array}{rl} 069312586011000000000000 & acx^2 \\ 62321861362000000 & (ad+bc)x \\ 003189603160 & bd \\ \hline 069312648332864551603160 & uv \end{array}$$

Au final, nous avons réduit une multiplication à 12 chiffres à trois multiplications à 6 chiffres et quelques additions et soustractions.

En général, multiplier des nombres deux fois plus longs prend environ trois fois plus de temps. Plus précisément, on a l'« inégalité récurrente » suivante :

$$\mathrm{M}(2n) \leqslant 3\,\mathrm{M}(n) + O(n).$$

Pour une certaine constante C et pour $n = 2^r$, nous avons donc

$$\begin{aligned} \mathrm{M}(n) &\leqslant 3\,\mathrm{M}(n/2) + Cn \\ &\leqslant 9\,\mathrm{M}(n/4) + \left(1+\tfrac{3}{2}\right)Cn \\ &\leqslant 27\,\mathrm{M}(n/8) + \left(1+\tfrac{3}{2}+\tfrac{9}{4}\right)Cn \\ &\vdots \\ &\leqslant 3^r\,\mathrm{M}(1) + O\left(\left(\tfrac{3}{2}\right)^r n\right) = O(3^r). \end{aligned}$$

Or nous pouvons toujours rajouter des zéros à gauche d'un nombre afin que son nombre de chiffres devienne une puissance de deux. En fin de compte, ceci prouve

$$\mathrm{M}(n) = O(n^{\log 3/\log 2}).$$

Des nombres aux polynômes

L'un des ingrédients de la méthode de Karatsuba est de couper les nombres en deux morceaux. Cela permet de voir u et v comme des polynômes $ax+b$ et $cx+d$ de degré <2. En fait, cela revient simplement à travailler en base x = 1000000 au lieu de 10.

L'idée de remplacer l'arithmétique entière par l'arithmétique polynomiale a déjà été suggérée par Kronecker au 19ème siècle. Par exemple, nous pouvons couper un nombre de n chiffres en l morceaux de $p := \lceil n/l \rceil$ chiffres et réinterpréter le résultat comme un polynôme.

Dans les années 1960, on a développé une série d'améliorations de la méthode de Karatsuba, en prenant le nombre de morceaux l supérieur à deux [24, 22, 19] ; voir le tableau 1

-4000	Onbekend	$O(n^2)$
1962	Karatsuba	$O(n^{\log 3/\log 2})$
1963	Toom	$O\left(n 2^{5\sqrt{\frac{\log n}{\log 2}}}\right)$
1966	Schönhage	$O\left(n 2^{\sqrt{\frac{2\log n}{\log 2}}} (\log n)^{\frac{3}{2}}\right)$
1969	Knuth	$O\left(n 2^{\sqrt{\frac{2\log n}{\log 2}}} \log n\right)$
1971	Pollard	$O(n \log n \log\log n \cdots)$
1971	Schönhage-Strassen	$O(n \log n \log\log n)$
2007	Fürer	$O(n \log n 2^{O(\log^* n)})$
2014	Harvey-vdH-Lecerf	$O(n \log n 8^{\log^* n})$
2017	Harvey	$O(n \log n 6^{\log^* n})$
2017	Harvey-vdH	$O(n \log n (4\sqrt{2})^{\log^* n})$
2018	Harvey-vdH	$O(n \log n 4^{\log^* n})$
2019	Harvey-vdH	$O(n \log n)$

Tableau 1. De meilleures limites supérieures pour $M(n)$ au fil des ans. Dans le tableau, la fonction $\log^*$ est définie par $\log^* x = \min\{n \in \mathbb{N} : l(\log \circ \overset{n\times}{\cdots} \circ \log)(x) \leqslant 1\}$.

pour un aperçu historique.

Sans entrer dans les détails, chacune de ces améliorations est basée sur une généralisation de l'astuce de Karatsuba. La multiplication de deux polynômes de degré $< l$ se réduit alors à $2l-1$ multiplications de coefficients, après quoi $M(n) = O(n^{\log(2l-1)/\log(l-1)})$.

Des polynômes aux cyclonômes

Désormais, nous nous focaliserons principalement sur la multiplication de polynômes et il est utile de travailler avec des coefficients dans un anneau général R. Nous laissons de côté le choix précis de R, pour l'instant.

Pour la suite, il est également utile de travailler avec des « cyclonômes » au lieu de polynômes. Un *cyclonôme* de degré l est un élément de $R[x]/(x^l-1)$.

La relation $x^l = 1$ n'entre en action que lorsque le degré d'un polynôme excède l. Le produit de deux polynômes de degré $< l/2$ dans $R[x]$ peut ainsi tout aussi bien se calculer dans $R[x]/(x^l-1)$.

L'avantage des cyclonômes est que nous avons de nouvelles astuces de calcul à notre disposition. Par exemple, supposons que $l = 2$ et que 2 soit inversible dans R. Alors il devient possible d'optimiser la méthode de Karatsuba : modulo x^2-1 on a

$$(a_1 x + a_0)(b_1 x + b_0) = \frac{\hat{a}_0 \hat{b}_0 - \hat{a}_1 \hat{b}_1}{2} x + \frac{\hat{a}_0 \hat{b}_0 + \hat{a}_1 \hat{b}_1}{2},$$

où

$$\begin{aligned} \hat{a}_0 &= a_0 + a_1 & \hat{b}_0 &= b_0 + b_1 \\ \hat{a}_1 &= a_0 - a_1 & \hat{b}_1 &= b_0 - b_1. \end{aligned}$$

Dans $R[x]/(x^2-1)$, le calcul d'un produit ne nécessite donc que deux multiplications dans R. Il y a aussi un certain nombre d'additions, de soustractions et de divisions par deux.

Les relations ci-dessus viennent de la factorisation

$$(x^2-1) = (x-1)(x+1) \tag{1}$$

et de l'application du théorème des restes chinois à celle-ci :

$$\begin{aligned} R[x]/(x^2-1) &\cong R[x]/(x-1) \times R[x]/(x+1) \\ \overline{a_1 x + a_0} &\mapsto (\overline{a_0 + a_1}, \overline{a_0 - a_1}) \end{aligned}$$

Cela permet de remplacer les calculs arbitraires dans $R[x]/(x^2-1)$ par des calculs dans $R[x]/(x-1) \times R[x]/(x+1) \cong R^2$.

La multiplication FFT

La factorisation (1) est valable pour chaque anneau R. Dans certains anneaux, le polynôme x^l-1 se scinde de la même manière en facteurs linéaires pour certains $l > 2$. Cela arrive dès que R admet un élément ω avec $\sum_{k=0}^{l-1} (\omega^j)^k = 0$ pour $j = 1, \ldots, l-1$. Un tel élément ω est appelé racine principale d'unité d'ordre l et conduit à la factorisation

$$x^l - 1 = \prod_{0 \leqslant k < l} (x - \omega^k).$$

Si l est également inversible dans R, alors le théorème des restes chinois donne :

$$R[x]/(x^l-1) \cong \prod_{0 \leqslant k < l} R[x]/(x-\omega^k).$$

De gauche à droite, cet isomorphisme est appelé *transformée de Fourier discrète* ou DFT. Pour tout $A \in R[x]$ et $0 \leqslant k < l$ on a $x = \omega^k$ et $P(x) = P(\omega^k)$ modulo $x - \omega^k$, donc

$$\mathrm{DFT}(\bar{A}) = (\overline{A(1)}, \overline{A(\omega)}, \ldots, \overline{A(\omega^{l-1})}).$$

L'algèbre $R[x]/(x-\omega^k)$ est à son tour isomorphe à R pour tout k, d'où

$$R[x]/(x^l-1) \cong R^l.$$

Or des calculs dans l'anneau R^l se font à moindre frais : une multiplication dans R^l équivaut par exemple à l multiplications dans R. Si nous disposions d'algorithmes efficaces à la fois pour la DFT et son inverse DFT^{-1}, alors cela donnerait une bonne méthode pour multiplier des cyclonômes $A, B \in R[x]/(x^l-1)$:

$$AB = \mathrm{DFT}^{-1}(\mathrm{DFT}(A)\,\mathrm{DFT}(B)).$$

Cela s'appelle la *multiplication FFT*.

La transformation de Fourier rapide

Mais comment calculer rapidement une telle DFT ? Nous avons déjà étudié le cas $l=2$. Plus généralement, la factorisation

$$x^{2l}-1 \;=\; (x^l-1)(x^l+1)$$

pour des degrés pairs $2l$ induit l'isomorphisme

$$R[x]/(x^{2l}-1) \;\cong\; R[x]/(x^l-1)\times R[x]/(x^l+1).$$

Si $A, B \in R[x]$ sont des polynômes de degré $<l$, alors cet isomorphisme envoie $A + B\,x^l$ vers $(A+B, A-B)$; pour la lisibilité, nous omettons désormais les barres des modulos. Calculer $A \pm B$ revient à additionner et soustraire l fois dans R. Pour l'isomorphisme dans l'autre sens, il faut rajouter $2l$ divisions par deux.

Si ω est une racine principale d'unité d'ordre $2l$, on a en outre l'isomorphisme suivant :

$$\begin{aligned} R[x]/(x^l+1) &\;\cong\; R[x]/(y^l-1) \\ \sum_{0\leqslant k<l} a_k x^k &\;\mapsto\; \sum_{0\leqslant k<l} (a_k\omega^k)\,y^k. \end{aligned}$$

Le calcul de cet isomorphisme se réduit à l multiplications par des puissances de ω. Ici, il est important de noter qu'une multiplication par une puissance de ω est parfois moins chère qu'une multiplication arbitraire dans R (voir plus bas).

En résumé, cela montre comment une DFT de longueur $2l$ peut être réduite à deux DFTs de longueur l. Si on écrit $\mathsf{F}(l)$ pour le temps qu'il faut pour calculer une DFT de longueur l, puis $\mathsf{T}_\pm$ pour le temps d'une addition ou d'une soustraction dans R, et T_ω pour le temps d'une multiplication par une puissance de ω, alors nous avons

$$\mathsf{F}(2l) \;\leqslant\; 2\,\mathsf{F}(l) + 2l\,\mathsf{T}_\pm + l\,\mathsf{T}_\omega.$$

En appliquant cette formule récursivement dans le cas où $l = 2^{\lg l}$ est une puissance de deux, nous obtenons

$$\begin{aligned} \mathsf{F}(l) &\;\leqslant\; 2\,\mathsf{F}(l/2) + l\left(\mathsf{T}_\pm + \tfrac{1}{2}\mathsf{T}_\omega\right) \\ &\;\leqslant\; 4\,\mathsf{F}(l/4) + 2l\left(\mathsf{T}_\pm + \tfrac{1}{2}\mathsf{T}_\omega\right) \\ &\;\vdots \\ &\;\leqslant\; (l/2)\,\mathsf{F}(2) + (\lg l - 1)\,l\left(\mathsf{T}_\pm + \tfrac{1}{2}\mathsf{T}_\omega\right) \\ &\;\leqslant\; l\lg l\left(\mathsf{T}_\pm + \tfrac{1}{2}\mathsf{T}_\omega\right). \end{aligned} \tag{2}$$

Pour la transformation inverse, il faut rajouter l divisions par l.

Intermezzo

Avant de continuer, c'est un bon moment pour quelques commentaires. La FFT a percé en 1965, après la publication de l'article de Cooley et Tukey [3]. Mais une méthode similaire avait déjà été décrite dans des travaux non publiés de Gauss [6, 14].

Figure 4. Volker Strassen (à gauche) et Arnold Schönhage (à droite).

Par ailleurs, nous avons vu qu'une multiplication de cyclonômes peut être réduite à deux DFT directes plus une DFT inverse. En 1970, Bluestein a noté qu'une DFT de longueur l peut également être réduite à un produit de cyclonômes de même degré l [2].

Pour expliquer cela, supposons pour simplifier que l est pair et que η est une racine principale d'unité d'ordre $2l$ avec $\eta^2 = \omega$. Pour des entiers j, k, on a alors

$$\eta^{(j+l)^2} = \eta^{j^2}\,\eta^{2l(j+l/2)} = \eta^{j^2}$$

et

$$\omega^{jk} = \eta^{j^2}\,\eta^{k^2}\,\eta^{-(j-k)^2}.$$

Le coefficient j-ième de la DFT d'un cyclonôme $u_0 + \cdots + u_{l-1}x^{l-1}$ vaut donc

$$\sum_{0\leqslant k<l} u_k\omega^{jk} \;=\; \eta^{j^2}\sum_{0\leqslant k<l} (u_k\eta^{k^2})\,\eta^{-(j-k)^2}.$$

Dans la somme à droite nous reconnaissons maintenant le produit de deux cyclonômes :

$$\begin{aligned} V &\;=\; \sum_{0\leqslant k<l} (u_k\eta^{k^2})\,x^k \\ W &\;=\; \sum_{0\leqslant k<l} \eta^{-k^2}\,x^k. \end{aligned}$$

Retour aux nombres entiers

En 1971, pas moins de trois méthodes apparurent pour appliquer la FFT à la multiplication des nombres entiers [21, 23]. Chacune de ces méthodes reposait sur un choix distinct de R.

Figure 5. John Pollard.

Le choix le plus naturel est de prendre $R = \mathbb{C}$ et $\omega = e^{2\pi i/l}$. Bien sûr, nous ne pouvons travailler qu'avec des approximations de nombres complexes dans notre cas. Pour multiplier des nombres de n chiffres, on peut prouver qu'il suffit de travailler avec une précision de $C \log n$ chiffres derrière la virgule pour une certaine constante $C > 0$. Cela signifie que nous pouvons travailler avec $l = O(n/\log n)$ blocs de $O(\log n)$ chiffres. En combinaison avec (2), ceci donne

$$M(n) = O(l \log l\, M(\log n)) = O(n\, M(\log n)).$$

En appliquant cette formule de manière récursive, nous obtenons

$$M(n) = O(n \log n \log\log n \log\log\log n \cdots).$$

Ceci donnait le «premier algorithme» de l'article de Schönhage–Strassen [23].

Cependant, l'algorithme zéro fut découvert légèrement plus tôt par Pollard [21]. Il proposa de prendre $R = \mathbb{F}_p$, où p est un nombre premier de la forme $p = s\,2^r + 1$ (plus que s soit petit, mieux que ça vaut). Pour de tels nombres premiers p, nous savons qu'il existe des racines primitives d'unité d'ordre 2^r. Cette fois-ci encore, cela permet de choisir $\log p = O(\log p)$ et $l = O(n/\log n)$, ce qui conduit à la même borne de complexité pour $M(n)$ que ci-dessus. Pour être tout à fait correct, il faut noter que cette borne de complexité ne figurait pas dans l'article de Pollard. Il était plus intéressé par un algorithme pratique et son article décrit plusieurs optimisations en ce sens. Pour de très grands nombres, son algorithme est toujours le meilleur sur les ordinateurs d'aujourd'hui [8].

Pour le «deuxième algorithme» de Schönhage–Strassen, nous passons au système binaire et prenons $R = \mathbb{Z}/(2^m + 1)\mathbb{Z}$, où m est une puissance de deux. Dans cet anneau, nous avons par définition $2^m = -1$, de sorte que $\omega := 2$ est une racine principale d'unité d'ordre $2m$. Un autre avantage est que ω est une racine « rapide » de l'unité. Nous entendons par là que nous pouvons rapidement multiplier par des puissances de ω : il suffit de décaler les bits du nombre en prenant soin d'utiliser la relation $2^m - 1$ lorsque l'on dépasse 2^m. Pour $l \in \{m, 2m\}$ Schönhage et Strassen montrent ensuite comment une multiplication dans $\mathbb{Z}/(2^{lm/2} + 1)\mathbb{Z}$ se réduit à l multiplications dans R. En écrivant $M'(m)$ pour le coût d'une multiplication dans R, cela donne

$$M'(lm/2) \leqslant l\,M'(m) + O(m\,l \log l). \tag{3}$$

Le terme $O(m\,l \log l)$ vient des DFTs, où on utilise le fait que $T_\omega = O(l)$ dans R. Le terme $l\,M'(m)$ vient de la multiplication « interne » dans $\prod_{k=0}^{l-1} R[x]/(x - \omega^k) \cong R^l$.

Réexprimé en fonction de n, l'inégalité (3) conduit *grosso modo* à la relation

$$M'(n) \leqslant 2\,n^{1/2} M'(n^{1/2}) + C\,n \log n, \tag{4}$$

pour une certaine constante C. On peut ensuite « dérouler » cette formule :

$$\begin{aligned} M'(n) &\leqslant 2\,n^{1/2} M'(n^{1/2}) + C\,n \log n \\ &\leqslant 4\,n^{3/4} M'(n^{1/4}) + 2\,C\,n \log n \\ &\leqslant 8\,n^{7/8} M'(n^{1/8}) + 3\,C\,n \log n \\ &\vdots \\ &\leqslant n \log n\, M'(O(1)) + C\,n \log n \log\log n. \end{aligned}$$

Cela prouve que $M(n) = O(n \log n \log\log n)$, et cette borne a tenu pendant quarante-cinq ans.

Et ensuite ?

Parmi les trois méthodes de 1971 que l'on vient de rappeler, la dernière présente un inconvénient majeur : puisque $l \leqslant 2m$, une multiplication de longueur n est « seulement » réduite à des multiplications de longueur $O(\sqrt{n})$ au lieu de $O(\log n)$. En revanche, contrairement aux deux autres méthodes, les DFTs ne coûtent presque rien. Cela fournit deux pistes d'amélioration.

Une première option est de réduire les coûts des DFTs à coefficients dans $\mathbb{C}$ ou $\mathbb{F}_p$. En 2007, Fürer fut le premier a appliquer cette stratégie avec succès [5]. Sa méthode conduit à une inégalité de la forme

$$\frac{M(n)}{n \log n} \leqslant K \frac{M(n')}{n' \log n'} + O(1), \quad n' = O((\log n)^2)$$

et la borne suivante pour $M(n)$:

$$\begin{aligned} M(n) &= O(n \log n\, K^{\log^* n}) \\ \log^* x &= \min\left\{k \in \mathbb{N} : \left(\log \circ \overset{k\times}{\cdots} \circ \log\right)(x) \leqslant 1\right\}. \end{aligned}$$

Fürer ne s'attarda pas sur le « facteur d'expansion » $K > 1$ précis. Depuis 2014, David Harvey, Grégoire Lecerf, et moi-même avons pu faire baisser ce facteur de plus en plus [13, 9, 10, 11] ; voir le tableau 1.

Notre deuxième option consiste à réexaminer l'inégalité (4). Or le facteur 2 dans $\underline{2}\, n^{1/2}\, \mathsf{M}'(n^{1/2})$ est très exactement compensé par le fait que $\log\sqrt{n} = \frac{1}{2}\log n$: en déroulant l'inégalité, chaque itération nécessite exactement $C\, n \log n$ opérations supplémentaires. Si nous pouvions améliorer ne serait-ce que très légèrement ce facteur 2, le déroulement se ferait ainsi :

$$\begin{aligned}
&\mathsf{M}(n)\\
&\leqslant 1.98\, n^{1/2}\, \mathsf{M}(n^{1/2}) + C\, n \log n\\
&\leqslant 1.98^2\, n^{3/4}\, \mathsf{M}(n^{1/4}) + (1 + 0.99)\, C\, n \log n\\
&\leqslant 1.98^3\, n^{7/8}\, \mathsf{M}(n^{1/8}) + (1 + 0.99 + 0.99^2)\, C\, n \log n\\
&\vdots\\
&\leqslant o(n \log n) + 100\, C\, n \log n.
\end{aligned}$$

Halte ! Nous avons bien lu ?

$$\mathsf{M}(n) = O(n \log n).$$

Avec cette nouvelle ligne de mire, nous allons pouvoir broder. Il suffit par exemple de prouver que

$$\mathsf{M}(n) \leqslant \alpha\, n^{(d-1)/d}\, \mathsf{M}(n^{1/d}) + O(n \log n), \tag{5}$$

où $n \geqslant 2$ et $\alpha < 1/d$.

En route vers les dimensions supérieures

De fait, la méthode de Schönhage et Strassen présente un aspect frustrant : la racine de l'unité $\omega = 2$ dans R, que nous avons créé artificiellement et à grand frais, est en réalité terriblement « sous-utilisée ». Pour cette raison, notre réduction de longueur $m \rightsquigarrow \sqrt{n}$ n'était que très modeste.

Or il existe aussi des DFTs qui fonctionnent dans plusieurs directions. Supposons à nouveau que R est un anneau arbitraire avec une racine d'unité ω d'ordre l. Une DFT d-dimensionnelle effectue l'isomorphisme

$$\begin{aligned}
&R[x_1, \ldots, x_d]/(x_1^l - 1, \ldots, x_d^l - 1)\\
&\cong\\
&\prod_{0 \leqslant k_1, \ldots, k_d < l} R[x_1, \ldots, x_d]/(x_1 - \omega_1^{k_1}, \ldots, x_d - \omega_d^{k_d})
\end{aligned}$$

Si nous remplaçons maintenant les DFTs en x_2,..., x_d par des DFTs à coefficients dans l'anneau $R[x_1]/(x_1^l - 1)$, ils coûteront beaucoup moins cher, puisque x_1 est une racine rapide de l'unité dans cet anneau. L'utilisation systématique de ce type de DFTs a d'abord été proposée par Nussbaumer et Quandalle [20].

Cette idée nous permet de faire un grand pas en direction de (5). Si $n = l^d$, alors une multiplication dans $R[x_1, \ldots, x_d]/(x_1^l - 1, \ldots, x_d^l - 1)$ nécessite $O(n \log n)$ additions et soustractions dans R plus $n^{(d-1)/d}$ multiplications dans $R[x_1]/(x_1^l - 1)$.

Il reste cependant un problème de taille : la multiplication rapide dans $R[x_1, \ldots, x_d]/(x_1^l - 1, \ldots, x_d^l - 1)$ n'est pas la même chose que la multiplication rapide dans $R[x]/(x^{l^d} - 1)$. Nous avons donc besoin d'un moyen de transformer des cyclonômes en une variable x en cyclonômes en plusieurs variables $x_1, \ldots, x_d$.

Avec l'aide de la Chine ancienne

Dans certains cas, il est en effet possible de changer de dimension. Supposons que $l_1, \ldots, l_d$ soient premiers entres eux. Selon le théorème des restes chinois, nous avons

$$\mathbb{Z}/(l_1 \cdots l_d)\mathbb{Z} \;\cong\; \mathbb{Z}/l_1\mathbb{Z} + \cdots + \mathbb{Z}/l_d\mathbb{Z}.$$

Formellement, ceci donne également

$$x^{\mathbb{Z}/(l_1 \cdots l_d)\mathbb{Z}} \;\cong\; x_1^{\mathbb{Z}/l_1\mathbb{Z}} \times \cdots \times x_d^{\mathbb{Z}/l_d\mathbb{Z}}.$$

Considérant ensuite des combinaisons linéaires, ceci nous amène enfin à

$$R[x]/(x^{l_1 \cdots l_d} - 1) \;\cong\; R[x_1, \ldots, x_d]/(x_1^{l_1} - 1, \ldots, x_d^{l_d} - 1).$$

En relation avec les DFTs, ceci a été remarqué pour la première fois par Good [7]. Agarwal et Cooley ont ensuite utilisé cet isomorphisme pour calculer des convolutions [1].

Cependant, nous avons maintenant un nouveau problème : dans la section précédente nous avions besoin d'un isomorphisme pour lequel tous les l_i étaient égaux. Mais notre isomorphisme ne marche que dans le cas où $l_1, \ldots, l_d$ sont au contraire premiers entre eux...

Le dernier ingrédient qui nous manque est un moyen de modifier légèrement la longueur d'une DFT. Cela permettrait de réduire une DFT d-dimensionnelle de longueur $(l_1, \ldots, l_d)$ en une autre de longueur $(l, \ldots, l)$. En utilisant une version d-dimensionnelle de l'algorithme de Bluestein, une telle DFT se réduit ensuite à une multiplication dans $R[x_1, \ldots, x_d]/(x_1^l - 1, \ldots, x_d^l - 1)$. Et cela peut être fait efficacement à l'aide des racines rapides de l'unité.

Rééchantillonnage gaussien

Comment remplacer une DFT de longueur s par une DFT de longueur t légèrement supérieure ? Pour y parvenir, nous supposons à partir de maintenant que $R = \mathbb{C}$.

Considérons une DFT de longueur s. Dans la théorie du signal, l'entrée est vue comme une série d'échantillons d'un signal. La fréquence d'échantillonnage est proportionnelle à s. Est-il possible de reconstruire le signal original à partir de notre ensemble d'échantillons ? Cela permettrait de prendre un nouvel ensemble d'échantillons, avec une fréquence différente.

La manière la plus évidente de rendre un signal numérique analogique est par convolution avec une gaussienne $G_\alpha(x) = \mathrm{e}^{-\alpha x^2}$. Plus α est petit, plus le signal analogique est lisse (mais moins net). Une propriété utile est que la transformée de Fourier de G_α est à nouveau une gaussienne.

Traduisons ces idées en formules. Au lieu de cyclonômes de degré s, nous considérons maintenant leurs vecteurs associés $u \in \mathbb{C}^s$ de coefficients. Nous convenons que $u_{k+js} \equiv u_k$ pour tout $j \in \mathbb{Z}$. Étant donné $u \in \mathbb{C}^s$, on définit $\mathscr{F}_s(u) \in \mathbb{C}^s$ par

$$(\mathscr{F}_s u)_k := \sum_{0 \le k < s} u_k \mathrm{e}^{-2\pi \mathrm{i} \frac{jk}{s}}.$$

Ceci est une variante de notre définition précédente d'une DFT (ici $\omega = \mathrm{e}^{-2\pi \mathrm{i}/s}$). Puis on définit deux applications linéaires $\mathscr{S}, \mathscr{T} : \mathbb{C}^s \longrightarrow \mathbb{C}^t$ par

$$\begin{aligned}(\mathscr{S} u)_k &:= \alpha^{-1} \sum_{j \in \mathbb{Z}} \mathrm{e}^{-\pi \alpha^{-2} s^2 (\frac{k}{t} - \frac{j}{s})^2} u_j \\ (\mathscr{T} u)_k &:= \sum_{j \in \mathbb{Z}} \mathrm{e}^{-\pi \alpha^2 t^2 (\frac{k}{t} - \frac{j}{s})^2} u_j.\end{aligned}$$

Enfin, nous introduisons deux permutations $\mathscr{P}_s : \mathbb{C}^s \longrightarrow \mathbb{C}^s$ et $\mathscr{P}_t : \mathbb{C}^t \longrightarrow \mathbb{C}^t$ par

$$\begin{aligned}(\mathscr{P}_s u)_j &:= u_{tj} \\ (\mathscr{P}_t u)_k &:= u_{-sk}.\end{aligned}$$

Dans [12, Theorem 4.2] nous montrons que le diagramme suivant commute :

$$\begin{array}{ccccc} \mathbb{C}^s & \xrightarrow{\mathscr{F}_s} & \mathbb{C}^s & \xrightarrow{\mathscr{P}_s} & \mathbb{C}^s \\ {\scriptstyle \mathscr{S}}\downarrow & & & & \downarrow{\scriptstyle \mathscr{T}} \\ \mathbb{C}^t & \xrightarrow{\mathscr{F}_t} & \mathbb{C}^t & \xrightarrow{\mathscr{P}_t} & \mathbb{C}^t \end{array}$$

C'est ce que nous utilisons pour réduire le calcul de $\mathscr{F}_s$ au calcul de $\mathscr{F}_t$.

Puisque $t > s$, les matrices de $\mathscr{S}$ et $\mathscr{T}$ ne sont pas carrées. Par construction, les éléments qui ne sont pas sur la diagonale diminuent rapidement. En effet, les gaussiennent déclinent à la vitesse de l'éclair loin du centre. En supprimant $t - s$ des lignes bien choisies de $\mathscr{T}$, il reste une matrice presque diagonale. Ceci peut être utilisé pour calculer rapidement $\mathscr{T}^{-1} \mathscr{P}_t \mathscr{F}_t \mathscr{S}$.

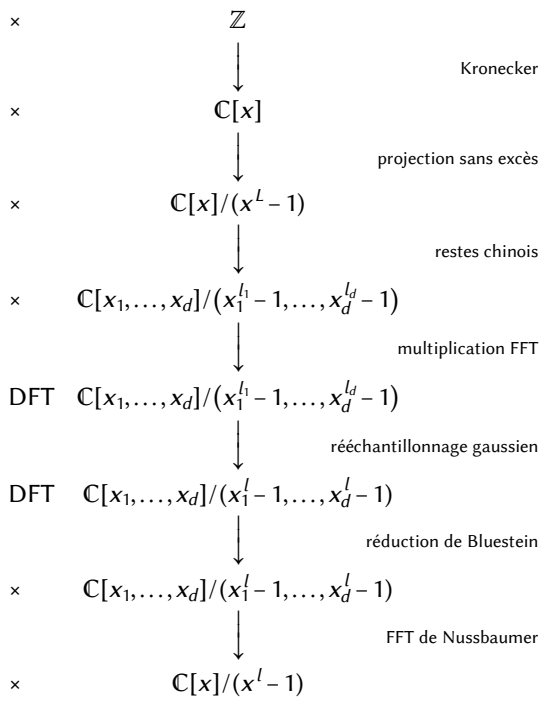

Figure 6. Représentation schématique des différentes réductions dans le nouvel algorithme. Nous avons triché un peu ici et là pour faire simple.

Si l'on choisit α et la précision de calcul avec précaution, on montre que $\mathscr{F}_s$ peut être calculé ainsi avec presque autant de précision que $\mathscr{F}_t$ et que le temps de calcul de $\mathscr{S}$ et de $\mathscr{T}^{-1}$ est négligeable par rapport à celui de $\mathscr{F}_t$.

Ceci parachève notre méthode et la démonstration que $\mathsf{M}(n) = O(n \log n)$. La figure 6 récapitule toutes les réductions que nous avons utilisées.

Une variante de notre méthode de rééchantillonnage fut publiée pour la première fois par Dutt et Rokhlin [4]. Cette variante est plus générale et permet de calculer des DFT quand échantillons des signaux sont irréguliers. En revanche, leur méthode ne fonctionne que pour $\log s = O(\alpha)$; de ce fait, elle est juste un peu trop lente pour notre application.

Et les applications ?

D'un point de vue pratique, nous verrons... Mais la fonction $\mathsf{M}(n)$ est importante pour la théorie, afin de décrire avec précision les coûts de toutes sortes d'opérations arithmétiques. Ainsi la division de deux entiers de $\leqslant n$ chiffres prend $O(\mathsf{M}(n)) = O(n \log n)$ opérations et le calcul d'un pgcd en prend $O(\mathsf{M}(n) \log n) = O(n \log^2 n)$. Désormais, on sait calculer

n chiffres de π en temps $O(M(n)\log n) = O(n\log^2 n)$. Une DFT complexe de longueur l nécessite $O(M(lp))$ opérations, si on calcule avec $p \geqslant \log l$ chiffres derrière la virgule [13]. Cela détermine également les émissions minimales de CO_2 pour des gros calculs sur le rechauffement climatique. D'une certaine manière, la fonction $M(n)$ comme « vitesse de l'arithmétique élémentaire » joue donc un rôle similaire à la vitesse de la lumière c en physique.

Existe-t-il des algorithmes plus rapides ?

Nous l'ignorons !

Bibliographie

[1] R. Agarwal et J. Cooley. New algorithms for digital convolution. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(5):392–410, 1977.

[2] Leo I. Bluestein. A linear filtering approach to the computation of discrete Fourier transform. *IEEE Transactions on Audio and Electroacoustics*, 18(4):451–455, 1970.

[3] J. W. Cooley et J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Computat.*, 19:297–301, 1965.

[4] A. Dutt et V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Comput.*, 14(6):1368–1393, 1993.

[5] M. Fürer. Faster integer multiplication. Dans *Proceedings of the Thirty-Ninth ACM Symposium on Theory of Computing (STOC 2007)*, pages 57–66. San Diego, California, 2007.

[6] C. F. Gauss. Nachlass: theoria interpolationis methodo nova tractata. Dans *Werke*, volume 3, pages 265–330. Königliche Gesellschaft der Wissenschaften, Göttingen, 1866.

[7] I. J. Good. The interaction algorithm and practical Fourier analysis. *Journal of the Royal Statistical Society*, Series B. 20(2):361–372, 1958.

[8] D. Harvey. Faster arithmetic for number-theoretic transforms. *J. Symbolic Comput.*, 60:113–119, 2014.

[9] D. Harvey. Faster truncated integer multiplication. https://arxiv.org/abs/1703.00640, 2017.

[10] D. Harvey et J. van der Hoeven. Faster integer and polynomial multiplication using cyclotomic coefficient rings. Technical Report, ArXiv, 2017. http://arxiv.org/abs/1712.03693.

[11] D. Harvey et J. van der Hoeven. Faster integer multiplication using short lattice vectors. Dans R. Scheidler et J. Sorenson, éditeurs, *Proc. of the 13-th Algorithmic Number Theory Symposium*, Open Book Series 2, pages 293–310. Mathematical Sciences Publishes, Berkeley, 2019.

[12] D. Harvey et J. van der Hoeven. Integer multiplication in time $O(n\log n)$. *Annals of Mathematics*, 193(2):563–617, 2021.

[13] D. Harvey, J. van der Hoeven, et G. Lecerf. Even faster integer multiplication. *Journal of Complexity*, 36:1–30, 2016.

[14] M. T. Heideman, D. H. Johnson, et C. S. Burrus. Gauss and the history of the FFT. *IEEE Acoustics, Speech and Signal Processing Magazine*, 1:14–21, oct 1984.

[15] J. van der Hoeven. Getallen vermenigvuldigen in $O(n\log n)$ stappen. *Nieuw Archief voor Wiskunde*, 21(1):55–60, 2020. Vijfde serie.

[16] J. van der Hoeven. *The Jolly Writer. Your Guide to GNU TeXmacs*. Scypress, 2020.

[17] A. A. Karatsuba. The complexity of computations. *Proc. of the Steklov Inst. of Math.*, 211:169–183, 1995. English translation; Russian original at pages 186–202.

[18] A. Karatsuba et J. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics Doklady*, 7:595–596, 1963.

[19] D. E. Knuth. *The Art of Computer Programming*, volume 2: Seminumerical Algorithms. Addison-Wesley, 1969.

[20] H. J. Nussbaumer et P. Quandalle. Computation of convolutions and discrete Fourier transforms by polynomial transforms. *IBM J. Res. Develop.*, 22(2):134–144, 1978.

[21] J. M. Pollard. The fast Fourier transform in a finite field. *Mathematics of Computation*, 25(114):365–374, 1971.

[22] A. Schönhage. Multiplikation großer Zahlen. *Computing*, 1(3):182–196, 1966.

[23] A. Schönhage et V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.

[24] A. L. Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. *Soviet Mathematics*, 4(2):714–716, 1963.