Université Paris Cité
UFR de Mathématiques et Informatique
45, rue des Saints-Pères, 75006, Paris.

Université Paris Cité

License 3ème année, Mathématiques et Applications, 2022-2023
TRANSFORMÉE DE FOURIER ET APPLICATIONS

Feuille de TD n°3

FFT : algorithme de la transformée de Fourier rapide.

Les exercices sont globalement indépendants mais nécessitent parfois des résultats vus dans les précédents. Ils sont pensés pour être faits dans l'ordre.

Exercice 1 (Produit polynomial)
Soit $A = a_0 + a_1X + \ldots + a_nX^n \in \mathbb{C}[X]$ et $B = b_0 + b_1X + \ldots + b_mX^m \in \mathbb{C}[X]$ deux polynômes complexes de degrés respectifs $n \in \mathbb{N}$ et $m \in \mathbb{N}$.

1. a) On note $C = AB$. Montrer que les coefficients de C notés c_ℓ, pour tout $\ell \in \{0, 1, \ldots, n+m\} \in \mathbb{C}^{n+m+1}$, vérifient

$$c_\ell = \sum_{k=0}^{\ell} a_{\ell-k} b_k.$$

b) Pourquoi ne peut-on pas écrire $c = (c_0, c_1, \ldots, c_{n+m})$ comme le produit de convolution $a * b$ où $a = (a_0, a_1, \ldots, a_n)$ et $b = (b_0, b_1, \ldots, b_m)$?

2. On pose $N = n + m + 1$ le nombre de coefficients du polynôme C. On note désormais a, b, c les éléments de ℓ_N définis par

$$\begin{aligned}(a(0), a(1), \ldots, a(N-1)) &= (a_0, a_1, \ldots, a_n, 0, \ldots, 0),\\ (b(0), b(1), \ldots, b(N-1)) &= (b_0, b_1, \ldots, b_m, 0, \ldots, 0),\\ (c(0), c(1), \ldots, c(N-1)) &= (c_0, c_1, \ldots, c_{N-1}).\end{aligned}$$

a) Montrer cette fois que

$$c = a * b.$$

b) Quelle est la complexité de ce calcul ?

3. On fixe $z_0, z_1, \ldots, z_{N-1}$ des nombres complexes deux à deux distincts.

a) Montrer que $\Phi : \mathbb{C}_{N-1}[X] \to \mathbb{C}^N$ définie par $\Phi(P) = (P(z_0), P(z_1), \ldots, P(z_{N-1}))$, pour tout polynôme P de degré au plus $N-1$, est un isomorphisme d'espaces vectoriels.

b) Montrer alors que $\Phi^{-1}(A(z_0)B(z_0), A(z_1)B(z_1), \ldots, A(z_{N-1})B(z_{N-1})) = AB$. En déduire une méthode alternative à celle présentée en 2)a) pour calculer le produit AB en passant par Φ.

c) Quelle est la complexité du calcul du vecteur $(A(z_0)B(z_0), A(z_1)B(z_1), \ldots, A(z_{N-1})B(z_{N-1}))$ à partir des valeurs $A(z_0), \ldots, A(z_{N-1})$ et $B(z_0), \ldots, B(z_{N-1})$? La méthode de la question précédente utilisée pour calculer le produit AB est-elle plus efficace que celle vue en 2)a) ?

Correction.

1. a) En développant, on remarque nécessairement que pour tout ℓ, $c_\ell = \sum_{j+k=\ell} a_j b_k$. Cette somme peut se réécrire en remplaçant j par $\ell - k$ et avec k variant de 0 à ℓ. D'où la réponse.

b) Même si l'expression des coefficients de c ressemble à un produit de convolution, celui-ci n'est défini qu'entre des éléments du même espace de suites périodiques. Or ici a et b ne sont pas définies comme des suites périodiques, mais surtout n'ont même pas le même nombre de coefficients. La question suivante règle ce problème.

2. a) On sait déjà que pour tout $\ell \in \{0, \dots, N-1\}$, $c(\ell) = \sum_{k=0}^{\ell} a(\ell-k)b(k)$. On déduit alors que

$$c(\ell) = \sum_{k=0}^{N-1} a(\ell-k)b(k),$$

car si $k > \ell$ alors forcément $a(\ell-k)b(k) = 0$ puisque :

—soit $\ell - k + N > n$, donc $a(\ell-k) = a(\ell-k+N) = 0$.

—soit $\ell - k + N \leqslant n$, donc $k \geqslant \ell + m + 1 > m$, d'où $b(k) = 0$,

On a donc bien $c = a * b$. Ainsi on a montrer que pour déterminer le produit de deux polynômes, il suffit de faire un zero-padding des vecteurs de leurs coefficients, puis les périodiser et enfin calculer leur produit de convolution.

b) La complexité de ce produit de convolution (et donc de la multiplication polynomiale via cette méthode) est en $O(N^2)$: il y a N produits et $N-1$ sommes par coefficients, et N coefficients.

3. a) Φ est linéaire, définie entre des $\mathbb{C}$-ev de même dimension N. Donc pour montrer que Φ est un isomorphisme d'ev, il suffit de vérifier que son noyau est trivial. Or c'est le cas car un polynôme de degré au plus $N-1$ admet au plus $N-1$ racines distinctes.

b) On a pour tout k, $A(z_k)B(z_k) = (AB)(z_k) = C(z_k)$. Comme Φ est un isomorphisme d'ev et que C est de degré au plus $N-1$, on déduit que $\Phi^{-1}(A(z_0)B(z_0), A(z_1)B(z_1), \dots, A(z_{N-1})B(z_{N-1})) = C = AB$.

Ainsi pour déterminer le produit AB, il suffit de choisir N points deux à deux distincts de $\mathbb{C}$, évaluer A et B en ces points, faire les produits des évaluations, puis appliquer Φ^{-1} (trouver l'unique polynôme interpolateur).

c) N produits donc $O(N)$.

Dans la représentation (équivalente par isomorphisme) des polynômes par leurs évaluations en suffisamment de points distincts, effectuer le produit polynomial est très simple : il suffit de faire le produit des évaluations. Et la complexité est donc linéaire ($O(N)$). Cependant pour bénéficier de ces avantages, il faut payer au préalable le coût de l'évaluation des polynômes qui est en $O(N^2)$. On ne fait donc pas mieux que la méthode par convolution. Mais ! Il se trouve que l'on peut faire ces évaluations de manière plus efficace que $O(N^2)$ en choisissant judicieusement les nombres complexes $z_0, z_1, \dots, z_{N-1}$. C'est l'algorithme FFT, qui donne une complexité en $O(N \log(N))$. Voir la suite...

Exercice 2 (DFT et polynôme)
Soit $a_0, a_1, \dots, a_{N-1} \in \mathbb{C}$ et $a \in \ell_N$ tel que pour tout $k \in \{0, \dots, N-1\}$, $a(k) = a_k$. On note $A = a_0 + a_1 X + \dots + a_{N-1}X^{N-1} \in \mathbb{C}_{N-1}[X]$ et $\alpha = e^{-\frac{2i\pi}{N}}$.

1. Rappeler la définition de DFT(a).

2. Exprimer DFT(a) en fonction de A et α. Commenter.

3. Retrouver, grâce à la question 3.a) de l'exercice 1, que DFT est un isomorphisme d'espaces vectoriels.

4. Quelle est la complexité du calcul de DFT(a) ?

5. En reprenant les notations de l'exercice précédent pour a, b, retrouver que DFT($a * b$) = DFT(a) · DFT(b).

Correction.

1. On a DFT(a) = $\hat{a} = (\hat{a}(m))_{m\in\mathbb{Z}} = (\langle a, \mathcal{E}_m \rangle)_{m\in\mathbb{Z}} = \left(\sum_{n=0}^{N-1} a_n e^{-\frac{2i\pi mn}{N}}\right)_{m\in\mathbb{Z}}$.

> Tout comme a, $\hat{a}$ est une suite N-périodique que l'on peut représenter de manière équivalente par le vecteur $(\hat{a}(m))_{0\leqslant m\leqslant N-1} \in \mathbb{C}^M$.

2. On a pour tout $m \in \mathbb{Z}$, $A(\alpha^m) = \sum_{n=0}^{N-1} a_n \left(e^{-\frac{2i\pi m}{N}}\right)^n = \hat{a}(m)$. Donc DFT($a$) = $(A(\alpha^m))_{m\in\mathbb{Z}}$. Ainsi déterminer la transformée de Fourier discrète de a revient à évaluer le polynôme de $\mathbb{C}_{N-1}[X]$, dont les coefficients (dans l'ordre croissant des degrés des monômes) sont les $a(0), a(1), \dots, a(N-1)$, en les α^m, $m \in \mathbb{Z}$.

3. Soit $w \in \ell_N$. D'après la question 3.a) de l'exercice 1, il existe un unique $P \in \mathbb{C}_{N-1}[X]$ tel que $P(\alpha^m) = w(m)$ pour tout $0 \leqslant m \leqslant N-1$, puisque les nombres complexes $1, \alpha, \ldots, \alpha^{N-1}$ sont deux à deux distincts. Notons $P = p_0 + p_1 X + \ldots + p_{N-1}X^{N-1}$, alors $p \in \ell_N$ défini par $p(n) = p_n$ pour tout $n \in \{0, 1, \ldots, N-1\}$ vérifie d'après la question précédente $\mathrm{DFT}(p) = (P(\alpha^m))_{m\in\mathbb{Z}}$, donc $\mathrm{DFT}(p) = w$. Ainsi DFT est surjectif et injectif (comme P est unique donc également p). Comme DFT est linéaire, c'est bien un isomorphisme d'ev.

4. On sait que pour déterminer $\mathrm{DFT}(a)$, il suffit de calculer les N coefficients de Fourier : $\hat{a}(0), \ldots, \hat{a}(N-1)$. Donc d'après ce qui précède il suffit d'effectuer les évaluations, $A(\alpha^0), \ldots, A(\alpha^{N-1})$. Chacunes demande $O(N)$ opérations, donc au total la complexité du calcul de $\mathrm{DFT}(a)$ par cette méthode est $O(N^2)$.

5. Pour A et B des polynômes de degrés respectifs n et m, on a vu dans l'Exercice 1 que $a * b$ représente la suite N-périodique dont les valeurs $a*b(0), a*b(1), \ldots, a*b(N-1)$ sont les coefficients du polynôme AB. Ainsi $\mathrm{DFT}(a*b) = ((AB)(\alpha^m))_{m\in\mathbb{Z}}$, et $((AB)(\alpha^m))_{m\in\mathbb{Z}} = (A(\alpha^m)B(\alpha^m))_{m\in\mathbb{Z}} = (A(\alpha^m))_{m\in\mathbb{Z}} \cdot (B(\alpha^m))_{m\in\mathbb{Z}} = \mathrm{DFT}(a) \cdot \mathrm{DFT}(b)$. D'où $\mathrm{DFT}(a*b) = \mathrm{DFT}(a) \cdot \mathrm{DFT}(b)$.

Exercice 3 (Fast Fourier Transform)
On suppose que $N = 2^p$ avec $p \in \mathbb{N}$. Soit $a_0, a_1, \ldots, a_{N-1} \in \mathbb{C}$ et $a \in \ell_N$ tel que pour tout $k \in \{0, \ldots, N-1\}$, $a(k) = a_k$. On note $A = a_0 + a_1 X + \ldots + a_{N-1}X^{N-1} \in \mathbb{C}_{N-1}[X]$ et $\alpha = e^{-\frac{2i\pi}{N}}$.

On souhaite trouver une méthode pour calculer $\mathrm{DFT}(a)$ de manière efficace (mieux que $O(N^2)$ opérations).

1. Rappeler le lien entre $\mathrm{DFT}(a)$ et A.

2. Montrer qu'il existe A_e et A_o des polynômes de degrés $\frac{N}{2} - 1 = 2^{p-1} - 1$ tels que $A(X) = A_e(X^2) + XA_o(X^2)$. On donnera leurs coefficients respectifs. *Indication : On pourra séparer les monômes composants A en fonction de leur parité.*

3. Montrer que pour tout $j \in \{0, 1, \ldots, \frac{N}{2} - 1\}$, on a

$$A(\alpha^j) = A_e(\alpha^{2j}) + \alpha^j A_o(\alpha^{2j}),$$
$$A(\alpha^{j+\frac{N}{2}}) = A_e(\alpha^{2j}) - \alpha^j A_o(\alpha^{2j}).$$

4. Que représente l'ensemble $S_{2^p} = \{\alpha^j, \alpha^{j+\frac{N}{2}} : j \in \{0, 1, \ldots, \frac{N}{2} - 1\}\}$?

5. Que représente l'ensemble $S_{2^{p-1}} = \{\alpha^{2j} : j \in \{0, 1, \ldots, \frac{N}{2} - 1\}\}$?

6. Représenter sur le cercle trigonométrique l'ensemble S_{2^3} et mettre en évidence l'ensemble S_{2^2}.

7. Supposons connues les valeurs $A_e(\alpha^{2j})$ et $A_o(\alpha^{2j})$ pour $j \in \{0, 1, \ldots, \frac{N}{2} - 1\}$. Montrer alors que l'on peut déterminer $(A(\alpha^0), A(\alpha), \ldots, A(\alpha^{N-1}))$ en $O(N)$ opérations.

8. On est donc ramené à déterminer $(A_e(\alpha^0), A_e(\alpha^2), \ldots, A_e(\alpha^{2N-2}))$ et $(A_o(\alpha^0), A_o(\alpha^2), \ldots, A_o(\alpha^{2N-2}))$. Expliquer comment on peut itérer le procédé précédent. Donner la complexité de cette nouvelle étape.

9. Jusqu'où peut-on répéter cette méthode ? L'ensemble de ces étapes correspond à l'algorithme FFT.

10. Evaluer la complexité totale de l'algorithme FFT.

11. Proposer une méthode pour calculer efficacement un produit polynomial à l'aide de la FFT.

Correction.

1. On a $\mathrm{DFT}(a) = (A(\alpha^m))_{m\in\mathbb{Z}}$.

> Dans la suite, nous ferons l'identification $\mathrm{DFT}(a) = (A(\alpha^m))_{0\leqslant m\leqslant N-1}$.

2. On a $A = a_0 + a_2X^2 + \ldots a_{2^p-2}X^{2^p-2} + X(a_1 + a_3X^2 + \ldots a_{2^p-1}X^{2^p-2})$. Ainsi en posant

$$A_e = a_0 + a_2X + \ldots a_{2^p-2}X^{2^{p-1}-1},$$
$$A_o = a_1 + a_3X + \ldots a_{2^p-1}X^{2^{p-1}-1},$$

on a bien $A = A_e + XA_o$ et A_e, A_o sont de degrés $2^{p-1} - 1 = \frac{N}{2} - 1$. Pour trouver A_e et A_o il suffit donc de séparer les coefficients de A en les coefficients pairs et impairs et de leurs associer de manière canonique le polynôme de degré $\frac{N}{2} - 1$.

3. Soit $j \in \{0, 1, \ldots, \frac{N}{2} - 1\}$, alors d'après ce qui précède $A(\alpha^j) = A_e(\alpha^{2j}) + \alpha^j A_o(\alpha^{2j})$ et $A(\alpha^{j+\frac{N}{2}}) = A_e(\alpha^{2j+N}) + \alpha^{j+\frac{N}{2}} A_o(\alpha^{2j+N})$. Or $\alpha^{2j+N} = \alpha^{2j}$ car $\alpha^N = 1$ et $\alpha^{j+\frac{N}{2}} = -\alpha^j$ car $\alpha^{\frac{N}{2}} = e^{-i\pi} = -1$. D'où le résultat demandé.

4. C'est l'ensemble des racines $N = 2^p$-ième de l'unité.

5. C'est l'ensemble des racines $\frac{N}{2} = 2^{p-1}$-ième de l'unité. En effet on a $\frac{N}{2}$ nombres complexes deux à deux distincts et $(\alpha^{2j})^{\frac{N}{2}} = \alpha^N = 1$. Ainsi on obtient les racines 2^{p-1}-ième de l'unité en prenant le carré des $\frac{N}{2}$ premières 2^p-ième racines de l'unité (en parcourant le cercle trigonométrique dans le sens trigonométrique). De manière équivalente il suffit de ne garder qu'une racine 2^p-ième de l'unité sur 2 en parcourant le cercle trigonométrique et en gardant en premier 1.

6.

7. Grâce aux deux expressions trouvées en 3), on détermine toutes les valeurs $A(\alpha^0), A(\alpha), \ldots, A(\alpha^{N-1})$ (donnant DFT(a)) par la seule connaissance des valeurs $A_e(\alpha^{2j})$ et $A_o(\alpha^{2j})$ pour $j \in \{0, 1, \ldots, \frac{N}{2} - 1\}$. Pour chacunes des équations en 3), on a une addition et une multiplication (on suppose les α^j précalculés), d'où une complexité totale de $2N$ soit $O(N)$.

8. On est donc ramené au calcul des $A_e(\alpha^{2j})$ et $A_o(\alpha^{2j})$ pour $j \in \{0, 1, \ldots, \frac{N}{2} - 1\}$. Or le vecteur $(A_e(1), A_e(\alpha^2), \ldots, A_e(\alpha^{2^{p-1}}))$ est la DFT du signal $\frac{N}{2}$-périodique dont les valeurs successives sont les coefficients de A_e, car $S_{2^{p-1}}$ est l'ensemble des racines $\frac{N}{2}$-ième de l'unité. On a le même résultat pour A_o. On peut donc répéter le processus précédent, des questions 2) et 3), mais cette fois les deux polynômes A_e et A_o.

Comme A_e, A_o sont de degrés $\frac{N}{2}$, la complexité de cette nouvelle étape est $O(\frac{N}{2}) + O(\frac{N}{2}) = O(N)$.

9. On peut répéter ce procédé jusqu'à obtenir des polynômes de degré 0. On se retrouve alors avec $N = 2^p$ polynômes constants. La complexité pour calculer la DFT associée à chacun de ces N polynômes est de 1, donc la complexité totale de cette dernière étape est de nouveau $O(N)$.

10. Quand on calcule la DFT de manière naïve, chacuns des coefficients demandent également une complexité en $O(N)$. Mais il faut répéter N fois cette tâche, d'où $O(N^2)$. Ici le nombre d'étape dans cette algorithme est p avant d'obtenir les N polynômes constants. Or $p = \log_2(N)$. Ainsi on se retrouve avec une complexité totale de $O(N \log(N))$. Par exemple si $N = 2^{20}$ (soit environ 10^6) alors la complexité de la méthode naïve pour calculer la DFT exige environ 10^{12} opérations. A l'inverse, l'algorithme FFT exige environ $10^6 \times 20$ opérations soit de l'ordre de 10^7. On a besoin de 10^5 fois moins d'opérations ! C'est une amélioration considérable. Elle vient des propriétés de symétrie des racines de l'unité qui permettent à chaque étape de *diviser* par 2 le degré des polynômes, plutôt que de les diminuer d'une unité (ce qui aurait mené de nouveau à une complexité en $O(N^2)$).

11. Pour simplifier, on considère deux polynômes de degré au plus $N/2 - 1$ avec N une puissance de 2. En prenant la DFT de ces polynômes, on obtient leurs représentations équivalentes via les évaluations en des nombres complexes distincts. Par l'algorithme FFT, cette étape a une complexité en $O(N \log(N))$. On multiplie alors les évaluations, la complexité est $O(N)$ (vu dans l'exercice 1). Enfin on applique l'IDFT pour obtenir le polynôme produit. On sait que l'IDFT revient grosso modo à appliquer une DFT (car $W_N^{-1} = \frac{1}{N} W_n^*$) donc la complexité de cette dernière étape est de nouveau en $O(N \log(N))$ grâce à l'algorithme FFT. On se retrouve avec une complexité totale pour effectuer ce produit polynomial en $O(N \log(N)) + O(N) + O(N \log(N)) = O(N \log(N))$. C'est donc encore une fois nettement mieux que l'algorithme donné par la convolution vu à l'exercice 1 et qui a une complexité en $O(N^2)$.