Algorithmique numérique (LU3IN028)

TD-TME n°4 - Transformée de Fourier Discrète

1. TD

Exercice 1 (Pratique de la FFT)**.**

1. Quelle est la somme des racines n-ième de l'unité ?
2. Quel est leur produit si n est pair et si n est impair ?
3. Quelle est la FFT de $(1,0,0,0)$? Quel est la valeurs de ω dans ce cas ? De quel vecteur $(1,0,0,0)$ est-il la FFT ?
4. Trouver l'unique polynôme de degré 4 qui prend les valeurs $p(1) = 2$, $p(2) = 1$, $p(3) = 0$, $p(4) = 4$ et $p(5) = 0$.

Exercice 2 (Multiplication de polynômes par la FFT)**.** On veut multiplier les deux polynômes $x+1$ et x^2+1 via la FFT. Trouver une puissance de 2 appropriée, trouver la FFT des 2 suites, multiplier le résultat composante par composante et calculer la FFT inverse pour obtenir le résultat.

Exercice 3 (FFT entière)**.** La FFT vue en cours utilise les nombres complexes. Du fait de la précision finie des calculs sur les nombres flottants, les erreurs d'arrondi peuvent dégrader la qualité du résultat. Pour certains problèmes, on sait que la solution est entière et il est alors intéressant d'utiliser une variante de la FFT basée sur l'arithmétique modulaire afin de garantir l'exactitude des calculs. C'est par exemple le cas lorsque l'on veut multiplier des polynômes à coefficients entiers.

Soit n une puissance de 2 et k le plus petit entier tel que $p = kn+1$ soit premier. Soit g un générateur de $(\mathbb{Z}/p\mathbb{Z})^*$ et posons $\omega = g^k \pmod p$.

1. Montrer que ω est une racine primitive n-ième de l'unité dans $\mathbb{Z}/p\mathbb{Z}$.
2. Expliquer pourquoi la FFT et la FFT inverse sont bien définies quand ω est une racine primitive n-ième de l'unité dans $\mathbb{Z}/p\mathbb{Z}$.
3. Expliquer pourquoi la FFT et son inverse peuvent être calculées en $\mathcal{O}(n \log n)$ si on suppose que le cout des opérations sur les mots de longueurs $\mathcal{O}(\log n)$ est constant. On supposera que p et ω sont donnés.
4. Calculer la FFT modulo $p = 17$ du vecteur $(0,5,3,7,7,2,1,6)$.

Exercice 4 (FFT dans des anneaux)**.** Un élément ω d'un anneau A est une racine n-ième de l'unité si $\omega^n = 1$ et un tel élément est dit *primitif* si $1-\omega^i$ est inversible pour tout entier i avec $1 \le i \le n$.

1. Soit ω une racine n-ième de l'unité tel que $1-\omega^i$ ne soit pas un diviseur de zéro pour tout entier i avec $1 \le i \le n$. Montrer que les matrices de Vandermonde $M_n(\omega)$ et $M_n(\omega^{-1})$ vérifient
$$M_n(\omega)M_n(\omega^{-1}) = nI_n.$$
2. Soit ω une racine n-ième de l'unité. Déduire de la question précédente que ω est primitive si et seulement si $1-\omega^i$ est pas un diviseur de zéro pour tout entier i avec $1 \le i \le n$ et que n (vu comme $n1_A$) est inversible dans A.

3. Soit ω une racine n-ième de l'unité. Déduire de la question précédente que ω est primitive si et seulement si $1 - \omega^{n/p}$ est pas un diviseur de zéro pour tout p premier divisant n et que n (vu comme $n1_A$) est inversible dans A.
4. Soit ω une racine n-ième de l'unité, $k \in \mathbb{Z}$ et $d = n/\operatorname{pgcd}(n,k)$. Montrer que ω^k est une racine primitive d-ième de l'unité.

2. TME

Exercice 5 (FFT récursive et itérative)**.**

1. Implanter la version récursive de la FFT.
2. Implanter la version itérative de la FFT.
3. Comparer en terme d'efficacité vos deux implantations.
4. Comparer ensuite vos implantations avec la commande `fft` de MATLAB.

Exercice 6 (Compression d'images via la FFT)**.** La figure 1 représente une image de 320×200 pixels correspondante à une matrice X de taille 320×200.

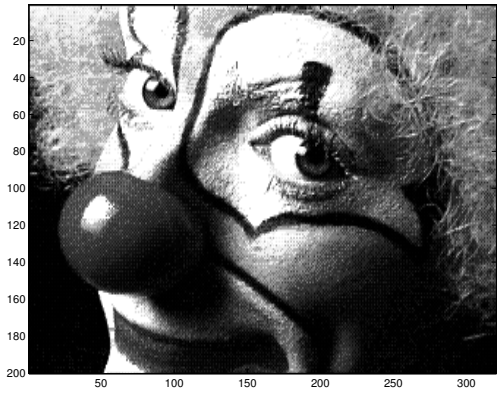

FIGURE 1 – Image de taille 320×200 pixels

L'affiche de l'image en MATLAB se fait de la façon suivante :

```
load clown.mat;
colormap('gray');
image(X);
```

On rappelle que l'on note

$$M_n(\omega) = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ & & \vdots & & \\ 1 & \omega^j & \omega^{2j} & \cdots & \omega^{(n-1)j} \\ & & \vdots & & \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{pmatrix}$$

avec $\omega = e^{2i\pi/n}$.

Comment compresser une image avec la FFT ? Après tout, une image est un tableau à 2 dimensions et jusqu'à maintenant nous n'avons parlé que de la FFT d'un tableau à une dimension. Soit X une matrice de taille $m \times n$ (ou une image). La FFT en 2 dimension consiste à calculer $ffX = M_m X M_n$. On peut implanter ce calcul de la façon suivante :

— pour chaque colonne de X, on calcul sa FFT. Notons fX la matrice obtenu. En d'autres termes, la colonne i de fX est la FFT de la colonne i de X.
— pour chaque ligne de fX, on calcule sa FFT. Noton ffX la matrice obtenue. En d'autres termes, la ligne i de ffX est la FFT de la ligne i de fX.

La matrice ffX est la FFT 2-dimensionnelle de X. On utilise ffX pour la compression de la manière suivante. On effectue la FFT de l'image mais on ne stocke que les éléments non nuls. En fait, pour compresser, on n'enlève pas seulement les zéros mais aussi tous les éléments dont la valeur absolue est plus petit qu'un seuil fixé ε.

1. En utilisant la FFT, proposer un algorithme de compression d'image. Tester votre algorithme sur la figure 1.

2. Calculer le taux de compression permettant de mesurer la qualité de compression des images.

Exercice 7 (Multiplication de polynômes)**.** Implanter l'algorithme vu en cours pour calculer de manière efficace la produit de deux polynômes. Vous tracerez le temps de calcul en fonction du degré des polynômes. Vous comparez avec la méthode classique de multplication de polynômes.