ESIR 1 / Algorithmique et Complexité : TP 2
Transformée de Fourier rapide (première partie)

Pierre Maurel, pierre.maurel@irisa.fr

1 Introduction

1.1 Objectif et contexte

L'objectif de ce TP est d'implémenter une version rapide de la transformée de Fourier discrète (TFD). Vous implémenterez une version récursive, mais sachez qu'il est possible d'implémenter une version itérative du même algorithme de manière plus efficace (même complexité mais "facteur multiplicatif" plus faible). Dans cette première séance de TP, nous verrons une première application de la TFD : la multiplication rapide de polynômes. Dans une deuxième partie, nous verrons une deuxième application : le traitement et la compression d'images.

1.2 Réalisation et données fournies

Ce TP se fera en `Java`. Vous pourrez utiliser `Eclipse` pour gérer votre programme. Un squelette de programme (`FFT_1D.java`) est disponible sur l'espace ENT du cours. Vous trouverez également un fichier `CpxTab.java`. Les objets de `CpxTab` représentent des tableaux de complexes. Vous regarderez le code pour identifier les différentes méthodes disponibles.

2 Implémentation récursive de la FFT

Étant donné un polynôme A représenté par ses coefficients $(a_0, \ldots, a_{n-1})$, on veut calculer sa transformée de Fourier discrète qui est égale au vecteur suivant :

$$(A(\omega_n^0), \ldots, A(\omega_n^{n-1})) \quad \text{avec} \quad \omega_n = e^{\frac{2\pi i}{n}} \text{ (racine n-ième principale de l'unité)}$$

On a montré en TD que, si on disposait de la TFD correspondante aux coefficients d'indices pairs de A et de celle correspondante aux coefficients d'indices impairs, on pouvait alors obtenir la TFD de A grâce aux formules suivantes :

$$\forall k \in \{0, \ldots, \frac{n}{2} - 1\} \ : \qquad \begin{cases} A(\omega_n^k) = A_{\text{pair}}(\omega_{n/2}^k) + \omega_n^k \, A_{\text{impair}}(\omega_{n/2}^k) \\ A(\omega_n^{k+\frac{n}{2}}) = A_{\text{pair}}(\omega_{n/2}^k) - \omega_n^k \, A_{\text{impair}}(\omega_{n/2}^k) \end{cases}$$

Exercice 1 Utilisez ces formules pour implémenter la fonction `CpxTab combine(CpxTab c1, CpxTab c2)` :

- `c1` correspond à la TFD des coefficients pairs, c'est à dire que `c1[k]` vaut $A_{\text{pair}}(\omega_{n/2}^k)$
- `c2` correspond à la TFD des coefficients impairs, c'est à dire que `c2[k]` vaut $A_{\text{impair}}(\omega_{n/2}^k)$
- La fonction doit donc renvoyer la TFD de A (c'est à dire un `CpxTab` contenant les $A(\omega_n^k)$) en supposant que les TFD correspondantes aux coefficients pairs et impairs de A sont déjà calculées et sont dans `c1` et `c2`.
- Vous pourrez utilisez les fonctions `Math.cos` et `Math.sin` de la bibliothèque java.lang.Math, ainsi que le `double Math.PI`.

Exercice 2 En utilisant la fonction `combine`, écrire une version récursive (`CpxTab FFT(CpxTab x)`) du calcul des coefficients de Fourier. Attention au test d'arrêt ! Vous testerez cette fonction sur l'exemple de l'exercice 5 du TD 3.

Exercice 3 Écrire la fonction `CpxTab FFT_inverse(CpxTab y)`, qui retrouve le signal d'entrée à partir de ses coefficients de Fourier. Vous vous servirez de la formule suivante :

$$TFD^{-1}(y) = \frac{1}{n}(TFD(y^*))^*, \tag{1}$$

où l'opérateur $*$ représente la conjugaison de nombres complexes. Vous testerez cette fonction au moins sur le résultat du test précédent.

3 Application à la multiplication polynomiale

Exercice 4 Compléter la fonction `multiplication_polynome_viaFFT` qui réalise la multiplication de polynômes en utilisant la transformée de Fourier, selon la méthode vue en en TD.

Vous testerez cette fonction au moins sur les exemples suivants : $A(X) = 2$ et $B(X) = -3$, $A(X) = 2 + X$ et $B(X) = -3 + 2\,X$.

La fonction `multiplication_polynome_viaCoeff` (donnée) réalise la multiplication de polynômes en passant par les coefficients (cf. exo 1 du TD).

Exercice 5 En utilisant le code présent dans le `main`. Comparez les temps d'exécution de ces deux méthodes (par les coefficients et par la FFT) pour différentes tailles de signal d'entrée (prenez des puissances de 2 uniquement). Vous appliquerez la même méthode d'analyse que lors du TP précédent. Que remarquez vous ?

Remarque : Comme dit en préambule, il est possible d'implémenter la FFT de manière impérative avec la même complexité que l'implémentation récursive mais de manière plus rapide (facteur multiplicatif plus petit).