Devoir 1 : Transformée de Fourier

Patrick Fournier

28 septembre 2020

Soit $(x_n)_{n=0}^{N-1}, x_n \in \mathbb{C}$ une suite de N nombres complexes. On définit sa transformée de Fourier (discrète) comme la suite de N nombres complexes $(X_k)_{k=0}^{N-1}$ telle que

$$X_k = \sum_{n=0}^{N-1} x_n \exp\left(-\frac{2\pi \imath k}{N} n\right) \tag{1}$$

où $\imath$ est l'unité imaginaire. Cette dernière est représentée par `i` dans R. Par exemple, les nombres complexes $4 + 3\imath$ et $2 + \imath$ s'écrivent `4 + 3i` et `2 + 1i` dans R.

Exercice 1

(a)

Implémentez naïvement (i.e. directement de la définition) eq. (1). Votre fonction doit

1. prendre deux argument, à savoir
   — un vecteur complexe x et
   — un nombre naturel k ;
2. retourner le nombre complexe correspondant X_k ;
3. faire appel à la fonction `sum` ;
4. s'appeler `dft1_naive`.

(b)

Implémentez une version itérative de eq. (1). Celle-ci doit respecter les exigences 1 et 2 de la fonction implémentée en (a). De plus, votre fonction doit

1. faire appel à une boucle `for` ;
2. s'appeler `dft1_iter`.

(c)

Implémentez une version matricielle de la eq. (1). Celle-ci doit respecter les exigences 1 et 2 de la fonction implémentée en (a). De plus, votre fonction doit

1. faire appel à la fonction `crossprod` ;
2. ne contenir qu'une seule ligne de code ;
3. ne pas contenir le symbole "`;`" ;
4. s'appeler `dft1_matrix`.

Notez que le nombre X_k n'est pas la même chose que la matrice 1×1 contenant X_k.

(d)

Implémentez une fonction permettant de calculer la transformée de Fourier d'une suite de nombres complexes $(x_n)_{n=0}^{N-1}$ à partir de l'une des fonctions (a), (b) ou (c). Votre fonction doit respecter les exigences 2 et 3 de (c). De plus, votre fonction doit

1. accepter deux arguments, à savoir
   — une fonction F et
   — un vecteur complexe x ;
2. retourner le vecteur complexe correspondant à $(X_k)_{k=0}^{N-1}$;
3. s'appeler `dft_factory`.

(e)

Implémentez les fonctions `dft_naive`, `dft_iter` et `dft_matrix` utilisant les fonctions `dft1_naive`, `dft1_iter` et `dft1_matrix` afin de calculer la transformée de Fourier d'une suite de nombres complexes $(x_n)_{n=0}^{N-1}$. Vos fonctions doivent respecter les exigences 2 et 3 de (c). De plus, vos fonctions doivent

1. accepter un seul argument, à savoir un vecteur complexe x ;
2. retourner le vecteur complexe correspondant à $(X_k)_{k=0}^{N-1}$.

Exercice 2

Une fonction récursive est une fonction contenant au moins un appel récursif, c'est-à-dire un appel à elle-même. Par exemple, on peut définir récursivement une fonction calculant la factorielle d'un nombre naturel ainsi :

```
fact <- function(x){
    x <= 0 && return(1)
    x * fact(x - 1)
}
> fact(5)
[1] 120
```

On définit $f_1, f_2, \ldots$ la suite de Fibonacci multiplicative positionnelle par

$$f_k = \begin{cases} k & \text{si } k \leq 2 \\ k f_{k-1} & \text{sinon} \end{cases}.$$

Implémentez `fib_mulPos_rec`, une fonction calculant *récursivement* le k^{e} terme de la suite de Fibonacci multiplicative positionnelle.

Exercice 3

Une des utilités des algorithmes récursifs est l'implémentation rapide et élégante de stratégies du type «diviser pour conquérir». Un exemple classique de ce type d'approche est l'algorithme de transformée de Fourier rapide de Cooley-Tukey en base 2. L'intuition est que pour N une puissance de 2 et $k = 0, \ldots, {}^{N}\!/_{2} - 1$, on a

$$\begin{cases} X_k &= P_k + \tau_k I_k \\ X_{k+N/2} &= P_k - \tau_k I_k \end{cases}$$

où

$$\tau_k = \exp\left(\frac{-2\pi \imath k}{N}\right)$$

et P_k et I_k sont la somme des termes pairs et impairs (à un facteur près) de eq. (1), i.e.

$$\begin{aligned} P_k &= \sum_{n=0}^{N/2-1} x_{2n} \exp\left(-\frac{2\pi\imath k}{N/2} n\right) \\ I_k &= \sum_{n=0}^{N/2-1} x_{2n+1} \exp\left(-\frac{2\pi\imath k}{N/2} n\right). \end{aligned}$$

Remarquez que P_k et I_k sont des transformée de Fourier d'une suite de nombres dont la longueur est une puissance de 2.
Implémentez un algorithme récursif exploitant cette relation en complétant la fonction ci-dessous. *Vous n'avez pas le droit de faire appel à une boucle `for` ou `while` dans votre solution.*

```
fft_ct2 <- function(x){
    ## Attention! Cette fonction suppose que la longueur
    ## de x est une puissance de 2!
    identical(ceiling(log2(length(x))), floor(log2(length(x)))) ||
        stop("La longueur de `x` doit être une puissance de 2")

    ## Condition d'arrêt.
    identical(length(x), 1L) && return(x)

    #####################
    ## À vous de jouer! ##
    #####################

    ## On calcule!

    ## On reconstruit le résultat.
}
```