Complément
Introduction à la FFT sur *MATLAB*

Simon Duval, Jean-Raphaël Carrier & Claudine Allen

La transformée de Fourier est une opération mathématique très souvent utilisée en recherche et en industrie. Dans les domaines de l'électronique et de l'optique, elle est d'une grande utilité pour traiter les signaux numériques. Une telle opération est généralement effectuée numériquement puisque les signaux considérés ne sont que très rarement représentés par des fonctions analytiques simples. On a donc recours à la forme discrète de la transformée de Fourier, communément appelée *Discrete Fourier Transform* (DFT). Cette dernière consiste à évaluer l'amplitude et la phase de chacune des composantes spectrales consistuant le signal. Évidemment, vu que le signal est discret, sa DFT sera également discrétisée. Cette discrétisation requiert d'abord une compréhension de base sur certains principes associés à l'échantillonnage d'un signal numérique.

1 Échantillonnage et fréquence de Nyquist

Pour un signal temporel quelconque mesuré avec une fréquence d'échantillonnage f_e, l'intervalle entre chaque mesure prise, c'est-à-dire l'intervalle d'échantillonnage Δt, est représenté par $\Delta t = 1/f_e$. La plus haute fréquence pouvant être échantillonnée, appelée *fréquence de Nyquist*, est donnée par :

$$f_N = \frac{1}{2\,\Delta t} = \frac{f_e}{2}. \tag{1}$$

Cette relation est d'une importance capitale en traitement de signaux numériques. Celle-ci nous indique qu'il faut choisir une fréquence d'échantillonnage suffisamment élevée pour représenter adéquatement un signal réel. Si le signal est mal échantillonné, ses fréquences élevées peuvent être interprétées à tort comme des basses fréquences. Ce phénomène appelé *repliement spectral* (en anglais : *aliasing*) peut causer une déformation du signal échantillonné et de sa transformée de Fourier.

2 La transformée de Fourier rapide (FFT)

Un algorithme très puissant et très rapide permet de calculer la DFT : la *Fast Fourier Transform* (FFT). L'algorithme permettant le calcul numérique de la transformée de Fourier inverse, appelé fort logiquement l'*Inverse Fast Fourier Transform* (IFFT), est identique à celui pour la FFT à un signe près. Il existe différentes méthodes pour calculer la FFT, mais la plus connue est celle introduite par Cooley et Tukey en 1865. Celle-ci consiste à calculer la DFT en plusieurs sous-DFT de tailles réduites. En général, la DFT est séparée en deux sous-DFT de même taille. Cette décomposition est répétée récursivement jusqu'à ce qu'il ne reste qu'un élément par DFT à effectuer. On peut ensuite remonter au résultat recherché à l'aide d'opérations arithmétiques simples entre chacun des résultats intermédiaires calculés. De ce fait, on inclut généralement cet algorithme dans la grande famille des algorithmes *Divide and Conquer* (diviser pour régner). Contrairement à l'implémentation directe de la DFT, qui requiert l'utilisation de N^2 opérations arithmétiques (N étant le nombre de points), la FFT en utilise seulement $N \log_2(N)$. Et puisque que les signaux numériques possèdent habituellement un grand nombre de points, le temps de calcul peut être significativement réduit.

3 La FFT et *MATLAB*

Le logiciel *MATLAB* possède des fonctions permettant le calcul de la FFT ou de la IFFT, appelées `fft.m` et `ifft.m`. Dans la plupart des cas, ces fonctions utilisent l'algorithme de Cooley et Tukey en divisant la DFT à calculer en $N/2$ DFT de même taille, de façon récursive. Pour diminuer le temps de calcul, il est donc préférable d'utiliser un nombre de points correspondant à une puissance de 2. Un nombre de points quelconque entré en argument oblige l'utilisation d'autres algorithmes de calcul de la FFT, qui sont généralement plus lents.

Considérons l'exemple d'une DFT à une dimension, représentée à la figure 1. Sur cette figure, on retrouve l'amplitude temporelle du signal dont on souhaite trouver la transformée de Fourier ainsi que le vecteur temps y étant associé. Ces vecteurs possèdent N points chacun et cette situation est adaptée au cas où N est une puissance de 2, donc un nombre pair. Pour cette raison, il y aura une asymétrie de part et d'autre du temps 0. Le vecteur temps peut donc être défini de la façon suivante, selon la syntaxe *MATLAB* :

$$t = \left[-\frac{N\Delta t}{2} : \Delta t : \frac{N\Delta t}{2} - \Delta t \right]. \tag{2}$$

On définit également le paramètre L, donné par :

$$L = N\Delta t. \tag{3}$$

Puisqu'il y a $N-1$ intervalles de temps, la largeur de la fenêtre temporelle est égale à $L - \Delta t$. La DFT d'un signal A_t sur *MATLAB* peut donc être effectuée de la façon suivante :

$$A_f = \mathrm{fft}(A_t)\,. \tag{4}$$

A_f est le vecteur associé à la DFT de A_t et est donc de même grandeur. De la même façon qu'il y a un temps associé à chaque élément du vecteur A_t, il y a aussi une fréquence associée à chaque élément du vecteur A_f généré. Cependant, la fonction `fft.m` ne donne pas l'option de générer ce vecteur fréquence pour l'utilisateur : il faut le générer par soi-même. Pour ce faire, comme le suggère la figure 1, certaines correspondances avec le vecteur temps doivent être respectées. Tout d'abord, la valeur absolue de la plus haute fréquence échantillonnée doit être égale à la fréquence de Nyquist pour respecter le théorème d'échantillonnage. Ensuite, le vecteur fréquence doit être de même grandeur que le vecteur temps et doit donc posséder N points. Ce vecteur doit également posséder à la fois des composantes en fréquences négatives et positives. À partir de ces règles, on peut déterminer l'intervalle entre deux fréquences consécutives du vecteur fréquence :

$$|F_{\max}| = f_N = \frac{1}{2\,\Delta t} = \frac{N}{2\,L}. \tag{5}$$

Or, on a également que :

$$f_N = \frac{N\,\Delta f}{2}. \tag{6}$$

En égalisant ces deux équations, on trouve que :

$$\Delta f = \frac{1}{L}. \tag{7}$$

Ce indique que la résolution en fréquence est, à peu de choses près, inversement proportionnelle à la largeur de la fenêtre temporelle, car $L \approx L - \Delta t$ si $\Delta t \ll L$. Le vecteur fréquence peut donc s'écrire selon la syntaxe *MATLAB* comme :

$$f = \left[-\frac{N\,\Delta f}{2} : \Delta f : \frac{N\,\Delta f}{2} - \Delta f\right]. \tag{8}$$

En termes du paramètre L, ce vecteur s'écrit comme :

$$f = \left[-\frac{N}{2\,L} : \frac{1}{L} : \frac{N}{2\,L} - \frac{1}{L}\right]. \tag{9}$$

Par convention, les algorithmes utilisés pour la FFT considèrent l'élément 1 du vecteur temps comme étant le temps 0 alors que dans notre cas, nous avons placé le temps 0 à la position $1+N/2$. Ceci a pour effet de décaler le vecteur A_f en fréquence, comme illustré à la figure 1. Il existe une fonction *MATLAB* permettant de redresser le vecteur A_f en fréquence. Celle-ci se nomme `fftshift.m`. Vous comprendrez qu'il existe également une fonction pour la IFFT, qui se nomme `ifftshift.m`. De plus, pour respecter le théorème de Parseval et obtenir une énergie dans le domaine des fréquences égale à celle dans le domaine temporel, il faut multiplier la DFT obtenue par l'intervalle d'échantillonnage. Voici comment correctement calculer A_f :

$$A_f = \text{fftshift}(\text{fft}((A_t))) \cdot \Delta t. \tag{10}$$

La IFFT peut être obtenue de façon similaire, sauf que l'`ifftshift` doit être effectué avant la fonction `ifft` et qu'on doit diviser le résultat par l'intervalle d'échantillonnage :

$$A_t = \text{ifft}(\text{ifftshift}(A_f)) \,/\, (\Delta t)\,. \tag{11}$$

Avec ces informations, vous devriez être en mesure d'évaluer la transformée de Fourier discrète de n'importe quel signal, qu'il soit réel ou complexe ! N'hésitez pas à vérifier votre code avec une fonction simple dont la transformée de Fourier analytique est facilement calculable. Par exemple, contruisez par vous-même une fonction rectangle (`rect.m`) de la forme suivante :

$$\text{rect}(x) = \begin{cases} 1 & \text{pour} \quad |x| < \frac{1}{2} \\ \frac{1}{2} & \text{pour} \quad |x| = \frac{1}{2} \\ 0 & \text{ailleurs} \end{cases} \tag{12}$$

La fonction temporelle peut donc être représentée par :

$$A_t = \text{rect}\left(\frac{t}{W}\right). \tag{13}$$

où W est la pleine largeur du rectangle. Comparez vos résultats avec la transformée de Fourier analytique de cette fonction rectangle, donnée par :

$$A_f = W\,\text{sinc}(W\,f) = W\,\frac{\sin(\pi\,W\,f)}{\pi\,W\,f}. \tag{14}$$

Puisque la transformée de Fourier produit généralement une fonction complexe, celle-ci est fréquemment représentée d'une part par sa norme ou sa norme au carré (fonction `abs.m`) et d'autre part par sa phase (fonction `angle.m`). N'oubliez pas de

choisir un vecteur temporel avec une largeur judieusement choisie et un nombre de points suffisant pour avoir une précision adéquate, et donc éviter les erreurs numériques ! Pour plus d'information, vous pouvez consulter le livre *Computational Fourier Optics : A MATLAB Tutorial* de David Voelz ; plusieurs exemples y sont présentés.

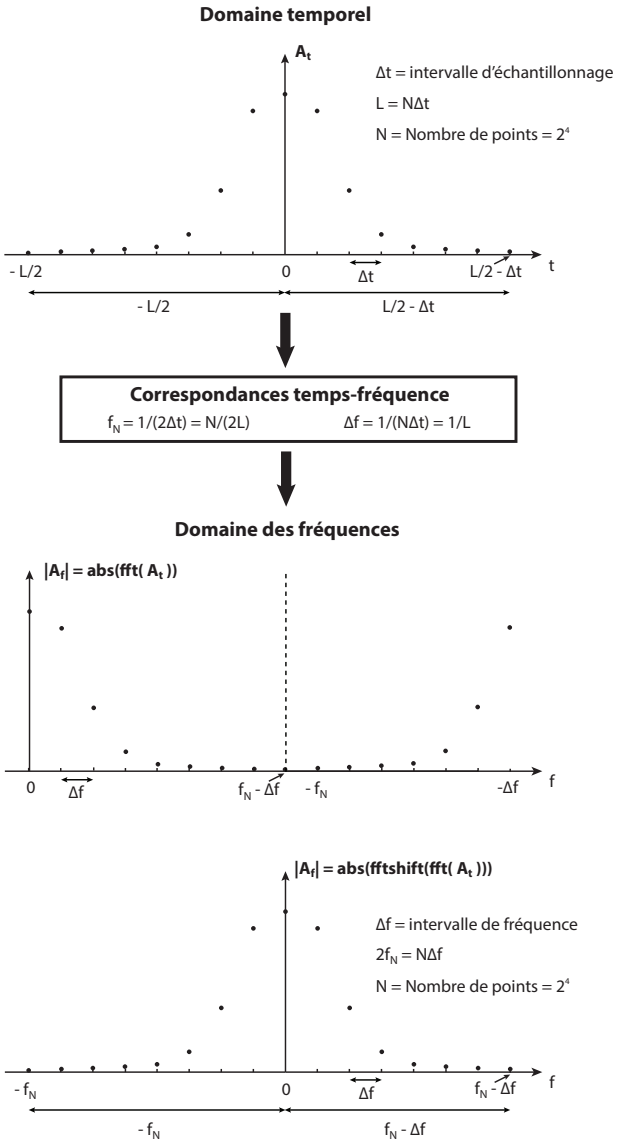

Figure 1 – Visualisation des concepts associés à la FFT sur *MATLAB*