Université de Toulouse III, FSI – Département de Mathématiques
Licence 2, module “Calcul scientifique”, 2019–2020

Projet: “Traque de fondamentales”

Février 2020

La transformée de Fourier discrète et ses variantes permettent d'analyser le contenu fréquentiel d'un signal. Lorsqu'on fait une telle analyse dans des cas simples, en particulier lorsqu'il s'agit de signaux temporels périodiques, on observe généralement la présence d'une *fréquence fondamentale* dans le signal, à laquelle viennent s'ajouter des multiples entiers appelés *harmoniques*. De nombreuses applications pratiques nécessitent de pouvoir détecter de manière fiable une fréquence fondamentale : accordeurs électroniques d'instruments de musique, analyse automatisée de morceaux de musique, analyse d'ondes gravitationnelles, etc.

Le but de ce projet est d'utiliser la transformée de Fourier discrète pour implémenter un analyseur de signal audio, capable de suivre la fréquence fondamentale du son au cours du temps.

Une expérience préalable de manipulation de transformées de Fourier discrètes (DFT et FFT) permettra de traiter plus rapidement le début de ce projet.

1 Signaux audio

Pour des raisons de simplicité de mise en œuvre, on se limitera à l'analyse de signaux audio. Un signal audio est une fonction $f\colon \mathbb{R} \to \mathbb{R}$, qui mesure la variation de pression de l'air au cours du temps par rapport à la pression d'équilibre. Pour des raisons pratiques évidentes, on considère la restriction de f à un intervalle compact $[0, T]$, et on la mesure en un nombre fini de points $(t_k)_{0\leq k\leq N}$ uniformément répartis. On obtient alors un *signal* $(u_k)_{0\leq k<N}$ défini comme l'échantillonnage de f aux points t_k :

$$\forall k \in \{0, \dots, N-1\}\colon u_k = f(t_k) \qquad\qquad t_k = k\frac{T}{N}. \tag{1}$$

Le réel $\frac{T}{N}$ s'appelle la *fréquence d'échantillonnage* du signal, et s'exprime en nombre d'échantillons par secondes ou en hertz. On manipulera en général des signaux audio échantillonnés à 44 100 Hz ou 48 000 Hz.

Il faut également garder à l'esprit que les valeurs mesurées lors de l'échantillonnage dépendent de capteurs, qui ont une amplitude de mesure finie. On supposera donc que les données ont été normalisées de manière à ce que f (et donc u) prennent toujours des valeurs comprises entre -1 et $+1$.

Pour manipuler des fichiers audio, on installera la bibliothèque audio `librosa` (voir listing 1). Elle permet d'extraire les données de fichiers audio (WAVE ou MP3) et de les obtenir sous forme d'un tableau `numpy` (voir les listings 2 et 3).

```
pip install librosa ffmpeg --user
```

Listing 1 – Installation de la bibliothèque audio

2 Transformée de Fourier discrète

La transformée de Fourier discrète (DFT pour *discrete Fourier transform*) est simplement un changement de base dans $\mathbb{C}^N$. Un signal $(u_k)_{0\leq k<N}$ peut être vu comme un vecteur de $\mathbb{C}^N$ (même si a priori ses composantes sont réelles). Pour correspondre aux conventions de numérotation des tableaux introduites par Python, on considérera (à la différence de l'indexation habituelle en mathématiques) que les composantes des vecteurs sont numérotées à partir de zéro.

2.1 Définition

On introduit la *base de Fourier* $(f_k)_{0\leq k<N}$ définie par :

$$f_k = \frac{1}{N}\begin{pmatrix} \exp\left(\dfrac{2i\times 0\times k\pi}{N}\right) \\ \exp\left(\dfrac{2i\times 1\times k\pi}{N}\right) \\ \exp\left(\dfrac{2i\times 2\times k\pi}{N}\right) \\ \vdots \\ \exp\left(\dfrac{2i\times (N-1)\times k\pi}{N}\right) \end{pmatrix} = \frac{1}{N}\begin{pmatrix} \omega^{0k} \\ \omega^{1k} \\ \omega^{2k} \\ \vdots \\ \omega^{(N-1)k} \end{pmatrix} \in \mathbb{C}^N \qquad \omega = e^{2i\pi/N} \in \mathbb{C}. \tag{2}$$

On vérifie aisément que $(f_k)_{0\leq k<N}$ forme une base orthogonale de $\mathbb{C}^N$ lorsqu'on munit ce dernier du produit scalaire hermitien canonique défini par

$$\forall u, v \in \mathbb{C}^N : u \cdot v = \sum_{k=0}^{N-1} u_k \overline{v_k}. \tag{3}$$

La *transformée de Fourier discrète* de u, qu'on notera $\hat{u} \in \mathbb{C}^N$, n'est rien d'autre que le vecteur des coordonnées de u dans cette nouvelle base, qui peuvent se calculer par produit scalaire d'après la remarque sur l'orthogonalité qu'on vient de faire :

$$\forall k \in \{0, \ldots, N-1\} : \hat{u}_k = \frac{u \cdot f_k}{\|f_k\|^2} = \sum_{n=0}^{N-1} u_n \omega^{-kn} = \sum_{n=0}^{N-1} u_n \exp\left(-\frac{2ikn\pi}{N}\right). \tag{4}$$

La matrice de passage de la base de Fourier à la base canonique s'écrit donc :

$$F = \begin{pmatrix} \omega^{-0\times 0} & \omega^{-0\times 1} & \ldots & \omega^{-0\times(N-1)} \\ \omega^{-1\times 0} & \omega^{-1\times 1} & \ldots & \omega^{-1\times(N-1)} \\ \vdots & & & \vdots \\ \omega^{-(N-1)\times 0} & \omega^{-(N-1)\times 1} & \ldots & \omega^{-(N-1)\times(N-1)} \end{pmatrix}. \tag{5}$$

On peut vérifier par calcul direct que cette matrice vérifie l'identité :

$$F^* F = N I_N. \tag{6}$$

Autrement dit la matrice adjointe de F (la conjuguée de sa transposée, notée F^*) est, à un facteur N près, son inverse, ce qui permet d'introduire la formule de reconstruction par la *transformée de Fourier inverse* :

$$u_n = \frac{1}{N}\sum_{k=0}^{N-1} \hat{u}_k \omega^{kn} = \frac{1}{N}\sum_{k=0}^{N-1} \hat{u}_k \exp\left(\frac{2ikn\pi}{N}\right). \tag{7}$$

2.2 Calcul en pratique : la FFT

Comme on l'a vu, N peut être très grand (de l'ordre de 10^5 pour un fichier audio de seulement quelques secondes). Le calcul direct de $\hat{u}$ à partir de la définition (4) a une complexité temporelle désastreuse, qui introduirait des temps de calcul trop élevés. On va donc considérer un algorithme de calcul récursif qui l'améliore considérablement, appelé *FFT* (fast Fourier transform).

Il est basé sur l'observation suivante. Si N est pair (en fait, on va considérer pour simplifier que c'est une puissance de 2) alors on peut décomposer le calcul de la somme définissant $\hat{u}$ en séparant les termes d'indices pairs et impairs :

$$\hat{u}_k = \sum_{n=0}^{N-1} u_n \exp\left(-\frac{2ikn\pi}{N}\right) \tag{8}$$

$$= \sum_{m=0}^{N/2-1} u_{2m} \exp\left(-\frac{2ik2m\pi}{N}\right) + \sum_{m=0}^{N/2-1} u_{2m+1} \exp\left(-\frac{2ik(2m+1)\pi}{N}\right) \tag{9}$$

$$= \sum_{m=0}^{N/2-1} u_{2m} \exp\left(-\frac{2ikm\pi}{N/2}\right) + \exp\left(-\frac{2ik\pi}{N}\right) \sum_{m=0}^{N/2-1} u_{2m+1} \exp\left(-\frac{2ikm\pi}{N/2}\right) \tag{10}$$

$$= \hat{v}_k + \omega^{-k} \hat{w}_k. \tag{11}$$

Ici, on a noté v et w les vecteurs obtenus en prenant respectivement les composantes paires et impaires de u :

$$v = (u_0, u_2, \ldots, u_{N-2}) \in \mathbb{C}^{N/2} \qquad w = (u_1, u_3, \ldots, u_{N-1}) \in \mathbb{C}^{N/2}. \tag{12}$$

Ainsi, pour calculer $\hat{u}$ (une transformée de taille N), il suffit de savoir calculer $\hat{v}$ et $\hat{w}$ (deux transformées de taille $N/2$). Cette remarque, combinée au fait que pour un vecteur de taille 1 on a $\hat{u} = u$ permet de donner un algorithme de calcul rapide de complexité $\mathcal{O}(N \log N)$.

Dans les cas où N n'est pas une puissance de 2, d'autres méthodes similaires faisant intervenir la décomposition de N en facteurs premiers existent. Il existe également des algorithmes de calcul efficace de la DFT pour les cas où N est un nombre premier, comme par exemple la *Chirp Z-transform* (CZT).

2.3 Interprétation de la DFT

Dans ce qui suit, on considère les fonctions « sinusoïdes complexes élémentaires » suivantes :

$$\forall F \in \mathbb{R}, \forall t \in \mathbb{R} \colon S_F(t) = \frac{1}{N} \exp\left(2i\pi F \frac{t}{T}\right). \tag{13}$$

Elles sont similaires aux fonctions de la base de Hilbert utilisée dans le contexte des séries de Fourier sur $L^2([0,T])$, à la différence que leur nombre d'oscillations F sur l'intervalle $[0,T]$ est un réel et non un entier. On notera par ailleurs que F peut être négatif.

Les vecteurs f_k de la base de Fourier peuvent être vus comme l'échantillonnage de certaines de ces sinusoïdes élémentaires complexes au sens de la définition (1) :

$$\forall k \in \left\{0, \ldots, N-1-\left\lfloor \frac{N}{2} \right\rfloor\right\} : f_k = \left(S_k\left(\frac{nT}{N}\right)\right)_{0 \leq n < N} \tag{14}$$

$$\forall k \in \left\{N-\left\lfloor \frac{N}{2} \right\rfloor, \ldots, N-1\right\} : f_k = \left(S_{k-N}\left(\frac{nT}{N}\right)\right)_{0 \leq n < N}, \tag{15}$$

où l'on a noté $\lfloor N/2 \rfloor$ la partie entière de $N/2$ pour le cas où N est impair. La valeur de F qu'on a choisie d'attribuer à f_k s'appelle un *mode*. Ainsi, les vecteurs $(f_k)_{0 \leq k < N}$ de la base de Fourier se voient attribués respectivement les *modes* suivants (voir figure 1) :

$$\left(0, 1, \ldots, N-2-\left\lfloor \frac{N}{2} \right\rfloor, N-1-\left\lfloor \frac{N}{2} \right\rfloor, -\left\lfloor \frac{N}{2} \right\rfloor, -\left\lfloor \frac{N}{2} \right\rfloor + 1, \ldots, -2, -1\right). \tag{16}$$

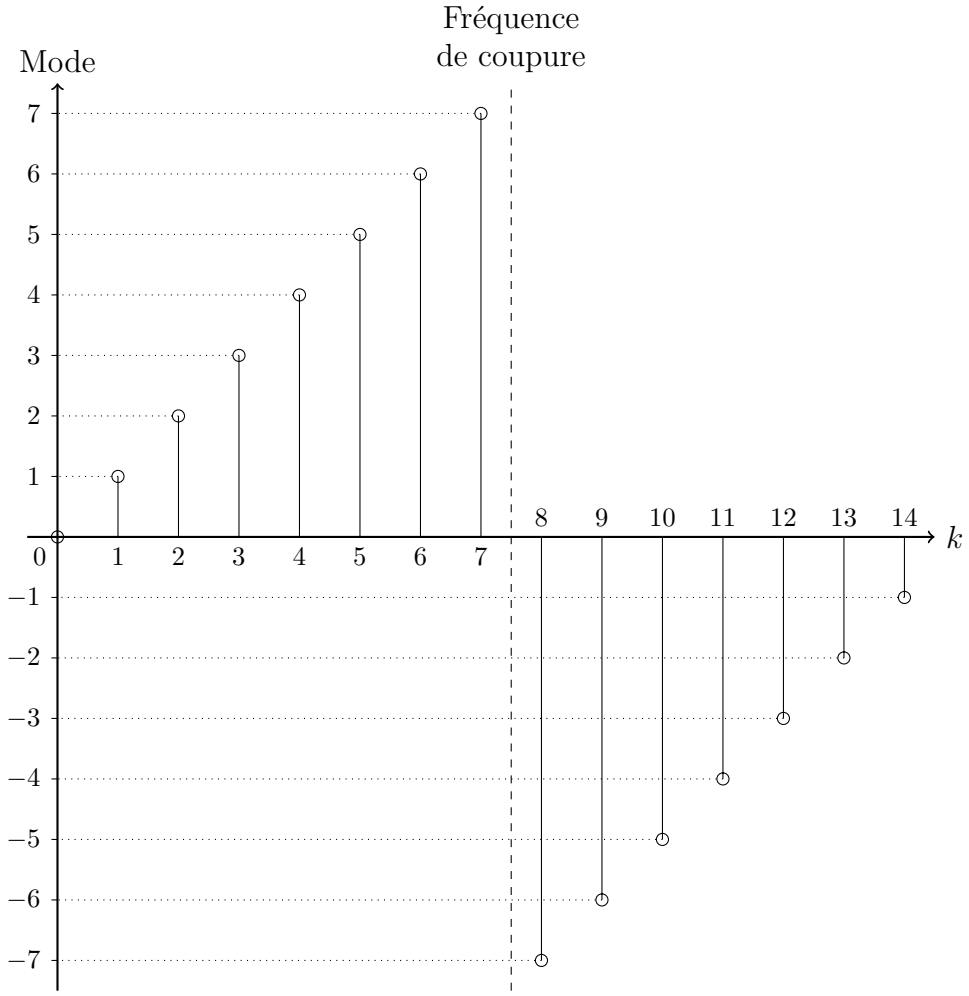

FIGURE 1 – Mode en fonction de l'indice dans la base de Fourier pour $N = 15$.

Le choix d'imposer une *fréquence de coupure* à la moitié du spectre (alors que d'autres choix auraient été possibles pour la correspondance entre le nombre d'oscillations et l'indice k) est délibéré et découle du théorème de Shannon, qui dit en substance que le contenu fréquentiel d'un signal échantillonné ne peut excéder la moitié de la fréquence d'échantillonnage.

Il faut bien comprendre que le mode est un entier sans dimension correspondant au nombre d'oscillations pendant la durée du signal. Pour déterminer la fréquence physique associée exprimée en hertz, il faut donc diviser le mode par la longueur temporelle du signal exprimée en secondes (voir figure 2).

Le vecteur $\hat{u}$ qu'on obtient lorsqu'on fait la DFT d'un signal u correspond aux coordonnées de ce signal dans la base de Fourier. On l'appelle le *spectre* du signal. Si l'une de composantes de $\hat{u}$ est non nulle, on dit que la fréquence correspondante (*bin* en anglais) est présente dans le spectre du signal.

```
import IPython.display as ipd
import numpy as np

T = 2
f = 441
sr = 44100
t = np.linspace(0, T, sr * T, endpoint = False)
u = np.sin(2 * np.pi * f * t)
ipd.Audio(u, rate = sr)
```

Listing 2 – Synthèse et écoute d'un signal sinusoïdal de 2 s à 441 Hz et échantillonné à 44 100 Hz

```
import librosa.core
import matplotlib.pyplot as plt

u, sr = librosa.core.load("violin1.mp3", None)
print("Fréquence d'échantillonnage :", sr, "Hz")
print("Durée du signal :", len(u) / sr, "secondes")
plt.plot(u)
```

Listing 3 – Ouverture d'un fichier MP3 et affichage des données brutes

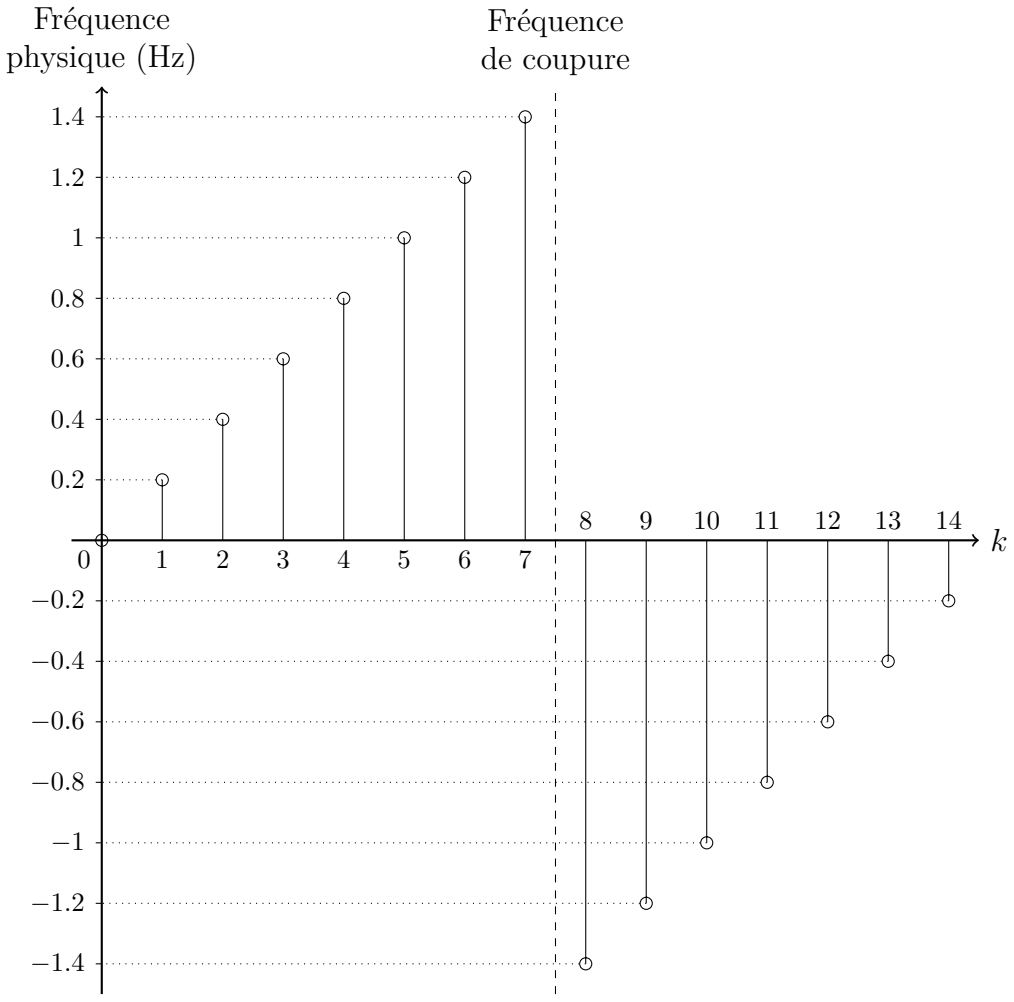

FIGURE 2 – Fréquence physique en fonction de l'indice dans la base de Fourier dans le contexte d'un signal de 5 secondes échantillonné à 3 Hz.

2.4 Représentation graphique de la DFT

Représenter directement la transformée de Fourier d'un signal u demande quelques précautions. Tout d'abord, étant donné que $\hat{u} \in \mathbb{C}^N$, on pourra considérer le module de $\hat{u}$: celui-ci donne des informations sur l'importance relative de chaque fréquence dans le spectre du signal. On obtient aussi généralement des images plus lisibles en passant en échelle logarithmique. Une manière simple de le faire est de représenter le log de $1 + |u_k|$ en fonction du mode (voir le listing 4).

On notera qu'en général, du fait que $\hat{u}_k = \overline{\hat{u}_{N-k}}$ lorsque le signal u est réel, le module est symétrique par rapport au centre du spectre.

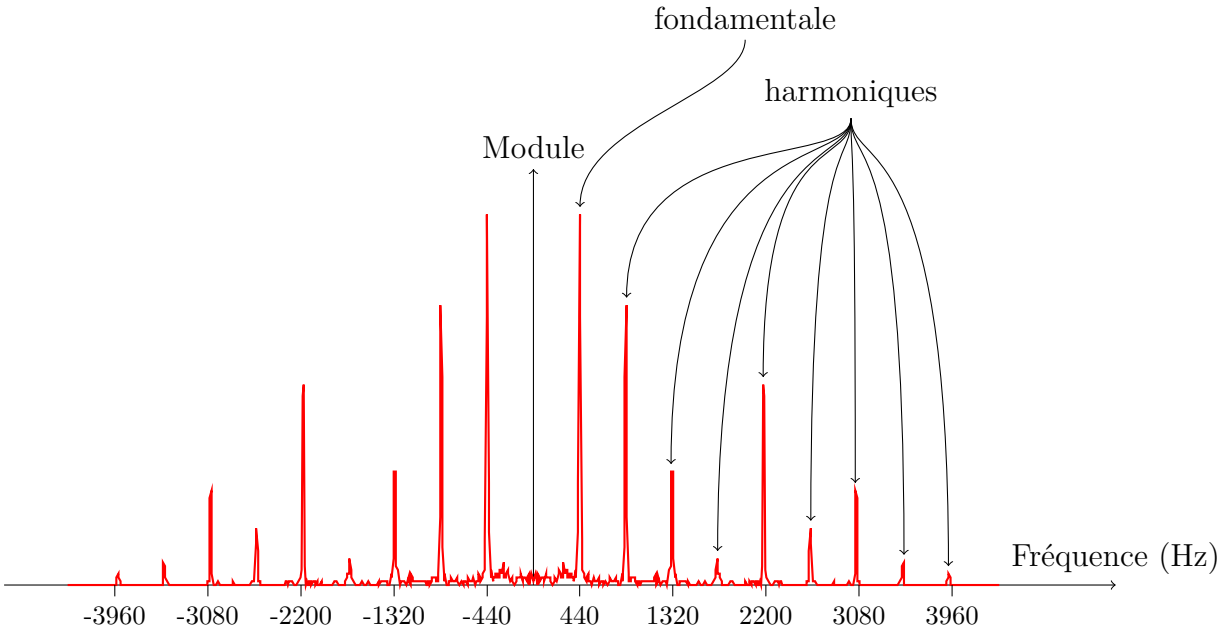

FIGURE 3 – Spectre en amplitude (module de la DFT) typique d'un instrument de musique à 440 Hz

2.5 Utilisation en Python

La bibliothèque `numpy` de Python fournit un certain nombre de fonctions pour manipuler la DFT :

- `numpy.fft.fft(u)` renvoie la DFT d'un tableau ;
- `numpy.fft.ifft(u)` renvoie la DFT inverse d'un tableau ;
- `numpy.fft.fftshift(u)` permute les deux moitiés d'un tableau, de chaque côté de la fréquence de coupure : ainsi on peut l'appliquer à un spectre de manière à le réoordonner par mode croissant ;
- `numpy.fft.ifftshift(u)` effectue l'opération inverse (c'est en général la même chose que `fftshift`, sauf si N est impair) ;
- `numpy.fft.fftfreq(N, 1 / sr)` renvoie les fréquences physiques, exprimées en Hz, qui correspondent aux coefficients de la DFT d'un signal de taille `N` lorsque la fréquence d'échantillonnage vaut `sr` (voir figure 2) ;

- `IPython.display.Audio(data, rate=sr)` permet de jouer le signal audio contenu dans un tableau `numpy` (voir listing 2). Il faut spécifier la fréquence d'échantillonnage `sr` ;
- `IPython.display.Audio(filename)` permet de jouer le signal audio contenu dans un fichier ;
- `librosa.core.load(filename, sr)` renvoie deux valeurs : le signal audio contenu dans un fichier et la fréquence d'échantillonnage (voir listing 3). Le paramètre `sr` peut prendre la valeur `None` (dans ce cas, la fréquence d'échantillonnage sera égale à celle prévue dans le fichier). Si on spécifie une fréquence d'échantillonnage différente, les données seront interpolées et rééchantillonnées à cette fréquence.

2.6 Localisation temporelle

Si l'on effectue directement la DFT d'un morceau de musique, le spectre résultant sera un mélange de toutes les fréquences des notes jouées et il sera difficile d'en extraire des informations pertinentes. Pour cette raison, on peut considérer seulement un sous-ensemble des données correspondant à une courte durée temporelle (par exemple 0,1 sec) dont on fait la DFT : on obtient ainsi le spectre « instantané » du signal. On appelle ces morceaux de signal des *fenêtres* (voir les figures 4 et 5).

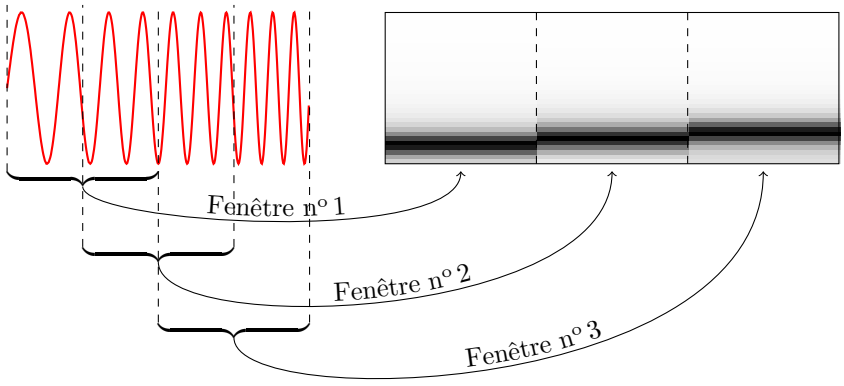

FIGURE 4 – Exemple de SDFT appliquée à un signal pseudo-sinusoïdal de fréquence croissante. On n'a représenté que la première moitié du spectre (pour $k < N/2$) car celui-ci est symétrique. On a pris $N = N'/2$ et $\mu = N/2$.

On parle alors de *transformée de Fourier discrète à fenêtre glissante* (SDFT pour sliding DFT). Explicitons cela. Si le signal u a une longueur totale $N' > N$ alors on peut définir la SDFT par :

$$\forall k \in \{0, \dots, N-1\}, \forall m \in \{0, \dots, N'-N\} : \ \mathrm{SDFT}(u)_{k,m} = \mathrm{DFT}\left((u_{m+n})_{0 \leq n < N}\right)_k \tag{17}$$

$$= \sum_{n=0}^{N-1} u_{n+m} \exp\left(-\frac{2i\pi kn}{N}\right). \tag{18}$$

On obtient ainsi une matrice complexe dont chaque colonne d'indice m est la DFT de la portion (fenêtre) de signal aux indices m à $m+N$. On notera que pour des considérations de vitesse de calcul, on se limite en général à calculer la SFDT seulement pour $m \in \{0, \mu, 2\mu, 3\mu, \dots\}$ pour un certain $\mu \in \mathbb{N}^*$ au lieu de toutes les valeurs de m possibles. La valeur de N' (nombre total d'échantillons du signal) étant fixée, voici les paramètres qu'on peut choisir en fonction du problème à traiter :

- la largeur de fenêtre $N \in \{1, \dots, N'-1\}$ détermine la « netteté » du spectre instantané : si elle est trop grande, la fenêtre risque de comporter plusieurs notes différentes et le spectre sera difficile à interpréter car contiendra un mélange de plusieurs notes. Mais si elle est trop petite, le spectre ne contiendra aucune information sur les hautes fréquences (on rappelle que le mode le plus grand s'arrête à $N/2$). Il faut donc trouver un compromis. Une valeur typique de N correspond à une fenêtre d'environ 0,1 sec, obtenue en prenant N égal au dixième de la fréquence d'échantillonnage du signal.

- le pas $\mu \in \mathbb{N}^*$ entre les fenêtres détermine la résolution temporelle de la SDFT. Plus il est petit, plus le nombre de fenêtres est élevé et plus on aura d'informations sur la variation locale du spectre. En contrepartie, les temps de calculs augmenteront. Une valeur typique est $\mu = N/2$, de manière à avoir un chevauchement à la moitié des fenêtres successives.

On pourra consulter le listing 5 pour un exemple d'implémentation du calcul et de l'affichage d'une SDFT.

3 Analyse du spectre

3.1 Détection approximative de la fondamentale

Une première idée pour détecter la fondamentale d'un signal consiste à déterminer

$$k_{\max} = \underset{0<k<N/2}{\arg\max} |\hat{u}_k|. \tag{19}$$

Toutefois, on risque en procédant ainsi de détecter une autre harmonique que la fondamentale. Cela est particulièrement vrai pour certains instruments (comme sur la figure 6) pour lesquels la fondamentale a une amplitude faible, voire nulle, comparée aux autres harmoniques.

Une méthode plus robuste consiste à prendre en compte dans la formule le poids de $J > 0$ harmoniques successives, et à pénaliser les fréquences intermédiaires :

$$k_{\max} = \underset{0<k<N/(2J)}{\arg\max} \left(\sum_{j=0}^{2J-2} (-1)^j \left| \hat{u}_{\lfloor k(1+j/2) \rfloor} \right| \right) \tag{20}$$

$$= \underset{0<k<N/(2J)}{\arg\max} \left(|\hat{u}_k| - \left| \hat{u}_{\lfloor 3k/2 \rfloor} \right| + |\hat{u}_{2k}| - \dots + |\hat{u}_{Jk}| \right). \tag{21}$$

On privilégiera ainsi la détection de motifs similaires au spectre idéal de la figure 3. On prendra typiquement $J \approx 6$.

3.2 Amélioration de la résolution fréquentielle par interpolation de l'amplitude

Comme on l'a vu, la DFT est de nature discrète. Cela a comme conséquence que les méthodes de détection de la fondamentale décrites plus haut ont une résolution fréquentielle relativement faible. Considérons par exemple un signal audio échantillonné à 44 100 Hz. On veut détecter la fondamentale avec une résolution temporelle de 0,1 sec. On fait donc une SDFT avec des fenêtres de longueur 4410. Le calcul de $k_{\max}$ donne une valeur comprise entre 1 et 2205 (correspondant respectivement à 10 Hz et 22 050 Hz) et la fréquence de fondamentale détectée ne sera donc précise qu'à 10 Hz près. Cela est très nettement insuffisant pour par exemple déterminer le nom d'une note de musique grave (autour de 50 Hz).

Pour améliorer la précision de la méthode, on peut interpoler $|\hat{u}|$ aux trois indices $k_{\max} - 1$, $k_{\max}$ et $k_{\max} + 1$ par un polynôme de degré 2 afin de déterminer une valeur réelle qui correspond

à son maximum. Cette méthode d'interpolation est facile à implémenter et donne d'assez bon résultats en pratique. On prendra garde toutefois à ne pas sortir de l'intervalle $[k_{\max}-1/2, k_{\max}+1/2]$ pour éviter des aberrations numériques qui peuvent se produire parfois.

3.3 Amélioration de la localisation fréquentielle par fenêtrage non rectangulaire

Soit $k_0 \in \{0, \dots, N/2\}$. Considérons pour commencer un signal sinusoïdal complexe de mode k_0 :

$$\forall n \in \{0, \dots, N-1\} \colon u_n = \exp\left(\frac{2ink_0\pi}{N}\right). \tag{22}$$

Alors on vérifie aisément que

$$|\hat{u}_k| = N\delta_{k,k_0} = \begin{cases} N & \text{si } k = k_0, \\ 0 & \text{sinon.} \end{cases} \tag{23}$$

Dans ces conditions, le spectre de u est très localisé et il est très facile de détecter la fondamentale : c'est l'unique composante non nulle de $\hat{u}$.

Il est toutefois très rare en pratique que la fréquence d'échantillonnage soit un multiple entier exact de la fréquence du signal. Pour formaliser ce type de situation, considérons cette fois une sinusoïde complexe de mode $k_0 + \varepsilon$:

$$\forall n \in \{0, \dots, N-1\} \colon u_n = \exp\left(\frac{2in(k_0+\varepsilon)\pi}{N}\right). \tag{24}$$

Ici on a pris $k_0 \in \{0, \dots, N/2\}$ et $\varepsilon \in]-1/2, 0[\cup]0, 1/2]$, de manière à ce que k_0 soit le mode (entier) le plus proche du mode (non entier) du signal. On peut alors calculer explicitement le module de la DFT de u, et tous calculs faits on trouve :

$$|\hat{u}_k| = \left|\frac{\sin(\pi\varepsilon)}{\sin\left(\dfrac{\pi(k_0+\varepsilon-k)}{N}\right)}\right|. \tag{25}$$

On constate bien à nouveau que $|\hat{u}_k|$ est maximal pour $k = k_0$. Toutefois, le spectre n'est pas nul au voisinage de la fondamentale. En utilisant l'inégalité :

$$\forall t \in [-\pi, \pi] \colon |\sin t| \leq \frac{|t(\pi - t)|}{\pi}, \tag{26}$$

pour majorer le dénominateur on en déduit que

$$|\hat{u}_k| \leq \left|\frac{N \sin \pi\varepsilon}{\left(\pi - \dfrac{\pi(k_0+\varepsilon-k)}{N}\right)}\right| \cdot \frac{1}{|\varepsilon + k_0 - k|}. \tag{27}$$

Au voisinage de k_0, on constate donc que $|\hat{u}_k|$ décroît comme $\frac{1}{|k_0-k|}$. On appelle ce phénomène une *fuite de spectre*. On va chercher à l'éviter au maximum, car en « érodant » les harmoniques du signal, il complique leur détection. On peut en voir une illustration sur la figure 7.

Une manière de limiter ce phénomène est de pré-multiplier le signal, avant de faire la DFT, par une sinusoïde (réelle cette fois) de mode $\frac{1}{2}$. On appelle cette étape préalable un *fenêtrage non rectangulaire*. Posons donc :

$$v_n = u_n \sin\left(\frac{n\pi}{N}\right). \tag{28}$$

On peut alors calculer

$$|\hat{v}_k| = \left| \frac{\cos\left(\pi(k_0+\varepsilon-k)\right)\sin\left(\frac{\pi}{N}\right)}{2\sin\left(\frac{\pi(k_0+\varepsilon+1/2-k)}{N}\right)\sin\left(\frac{\pi(k_0+\varepsilon-1/2-k)}{N}\right)} \right|, \tag{29}$$

et cette fois on constate une décroissance en $\frac{1}{|k_0-k|^2}$, ce qui est beaucoup mieux.

3.4 Amélioration de la résolution fréquentielle par interpolation de phase

Considérons l'exemple d'une sinusoïde complexe de mode $f' \in \mathbb{R}_+^*$. On se place cette fois dans le contexte d'une SDFT, avec $N' > N$:

$$\forall n \in \{0,\dots,N'-1\} \colon u_n = \exp\left(\frac{2inf'\pi}{N'}\right). \tag{30}$$

Sur chaque fenêtre de longueur N, le signal u se comporte comme une sinusoïde de mode $f = \frac{f'N}{N'} \in \mathbb{R}_+^*$. Pour ne pas dépasser la fréquence de coupure, on va supposer que $0 < f < \frac{N}{2}$. Comme on l'a fait précédemment, on peut prendre $k_0 \in \mathbb{N}$ et $\varepsilon \in]-1/2, 1/2]$ tels que $f = k_0+\varepsilon$, ou encore :

$$f' = \frac{fN'}{N} = \frac{(k_0+\varepsilon)N'}{N}. \tag{31}$$

Notons $(U)_{k,m}$ la SDFT de u et considérons deux colonnes de U d'indices m_1 et m_2 avec $m_2 > m_1$:

$$U_{k,m_1} = \sum_{n=0}^{N-1} u_{n+m_1} e^{-2i\pi kn/N} = \exp\left(\frac{2im_1f'\pi}{N'}\right) \sum_{n=0}^{N-1} \exp\left(2in\pi\left(\frac{f'}{N'} - \frac{k}{N}\right)\right) \tag{32}$$

$$U_{k,m_2} = \sum_{n=0}^{N-1} u_{n+m_2} e^{-2i\pi kn/N} = \exp\left(\frac{2im_2f'\pi}{N'}\right) \sum_{n=0}^{N-1} \exp\left(2in\pi\left(\frac{f'}{N'} - \frac{k}{N}\right)\right). \tag{33}$$

En supposant que U_{k_0,m_1} est non nul (ce qui est le cas d'après ce qu'on a vu précédemment), on constate que

$$\frac{U_{k_0,m_2}}{U_{k_0,m_1}} = \exp\left(\frac{2i(m_2-m_1)f'\pi}{N}\right), \tag{34}$$

d'où l'on tire que

$$\arg\left(U_{k_0,m_2}\right) - \arg\left(U_{k_0,m_1}\right) \equiv \frac{2(m_2-m_1)f'\pi}{N'} \qquad [2\pi], \tag{35}$$

ou encore, en remplaçant f' par son expression en fonction de k_0 et ε :

$$\varepsilon \equiv \frac{N\left(\arg\left(U_{k_0,m_2}\right) - \arg\left(U_{k_0,m_1}\right)\right)}{2\pi(m_2-m_1)} - k_0 \qquad \left[\frac{N}{m_2-m_1}\right]. \tag{36}$$

A priori, cette relation de congruence ne permet pas à elle seule de déterminer ε de manière unique. Toutefois, on se souvient qu'on a imposé $\varepsilon \in]-1/2, 1/2]$, en outre puisqu'on a nécessairement $N > m_2 - m_1$, on a $\frac{N}{m_2-m_1} > 1$. Dès lors, pour connaître ε, il suffit de calculer le membre de droite, et d'ajouter ou soustraire un certain nombre de fois $\frac{N}{m_2-m_1}$ pour être le plus proche possible de zéro : on obtient alors l'unique valeur possible de ε dans l'intervalle $]-1/2, 1/2]$.

On est maintenant en mesure de donner un algorithme de calcul précis de la fondamentale f' du signal u :

- Calculer la SDFT U du signal u avec un pas μ.
- Pour chaque colonne de U :
  - déterminer k_0 à l'aide de l'algorithme de maximisation décrit dans la section **3.1** ;
  - déterminer ε à l'aide de la relation **(36)**, en prenant pour m_1 la position de la fenêtre correspondant à la colonne courante et $m_2 = m_1 + \mu$ celle de la colonne suivante ;
  - une fois k_0 et ε connus, on en déduit la valeur de $f' = \frac{(k_0+\varepsilon)N'}{N}$: c'est le mode de la fréquence fondamentale instantanée du signal sur la fenêtre de SDFT considérée.

Cette méthode (dont a priori on sait seulement qu'elle fonctionne avec des sinusoïdes pures) améliore la précision fréquentielle de la fondamentale détectée dans la plupart des applications pratiques, en particulier dans les basses fréquences.

4 Fichiers fournis

L'ensemble des fichiers de travail peut être téléchargé à cette adresse :

`https://www.math.univ-toulouse.fr/~vfeuvrie/l2spe/fourier.zip`

Vous trouverez dans cette archive quelques fichiers audio pour vos expériences numériques :

- un fichier `violin1.mp3` avec trois notes de violon successives jouées avec un vibrato (la 440 Hz, do♯ 554 Hz et mi 659 Hz) ;
- un fichier `violin2.mp3` avec une gamme sur deux octaves jouée très rapidement (et qui présente une difficulté élevée pour la reconnaissance de la fondamentale) ;
- un fichier `oboe.mp3` avec une note de hautbois à 440 Hz ;
- un fichier `voice.mp3` avec sept notes chantées.

Il est également tout à fait possible d'apporter vos propres fichiers audio, pourvu qu'on y entende un instrument jouant au maximum une seule note à la fois. La détection de plusieurs fondamentales simultanées est un problème beaucoup plus ardu !

```python
# Chargement des données, avec rééchantillonnage à 4410 Hz pour un affichage
    plus lisible
u, sr = librosa.core.load("violin1.mp3", 4410)

# Extraction de la deuxième seconde du signal
v = u[1 * sr: 2 * sr]

# Calcul de la DFT
dft = np.fft.fft(v)

# Calcul des fréquences
frequencies = np.fft.fftfreq(len(dft), 1 / sr)

# Réordonnement par mode croissant des données
x = np.fft.fftshift(frequencies)
y = np.fft.fftshift(dft)

# Affichage
plt.figure(figsize = (10, 10))
plt.plot(x, np.log(1 + np.abs(y)))
plt.show()
```

Listing 4 – Représentation graphique d'une DFT en Python

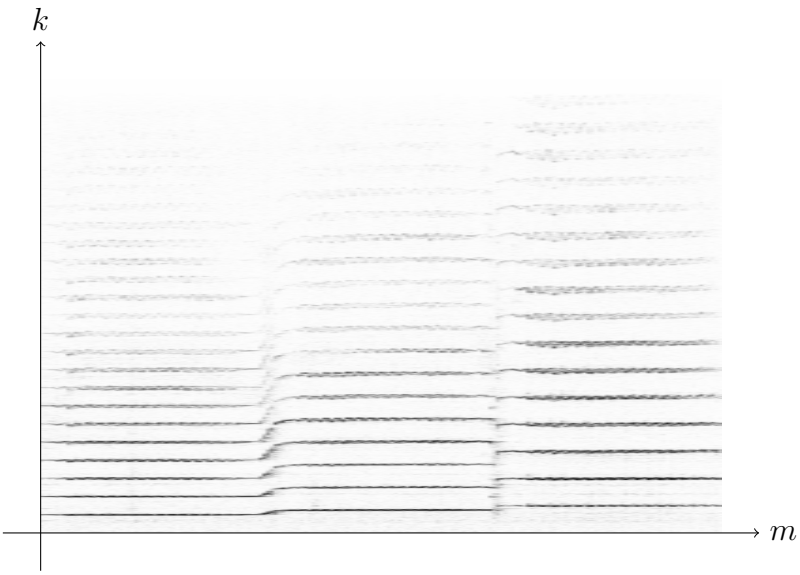

FIGURE 5 – SDFT (en module) d'un instrument de musique jouant successivement trois notes de fréquences croissantes. On peut observer les lignes horizontales correspondant aux harmoniques.

```python
def sdft(u, N, mu):
    M = 1 + (len(u) - N) // mu
    U = np.zeros((N, M), complex)
    for m in range(M):
        U[:, m] = np.fft.fft(u[m * mu: m * mu + N])
    return U

# Chargement des données
u, sr = librosa.core.load("violin1.mp3")

# Largeur de fenêtre correspondant à 0.1 sec de signal
N = sr // 10

# Espacement entre les fenêtres
mu = N // 2

# Calcul de la SDFT
s = sdft(u, N, mu)

# Affichage
plt.figure(figsize = (10, 10))
plt.imshow(np.log(1 + np.abs(s[:N // 2, :])), aspect = "auto")
plt.show()
```

Listing 5 – Calcul et affichage d'une SDFT en Python

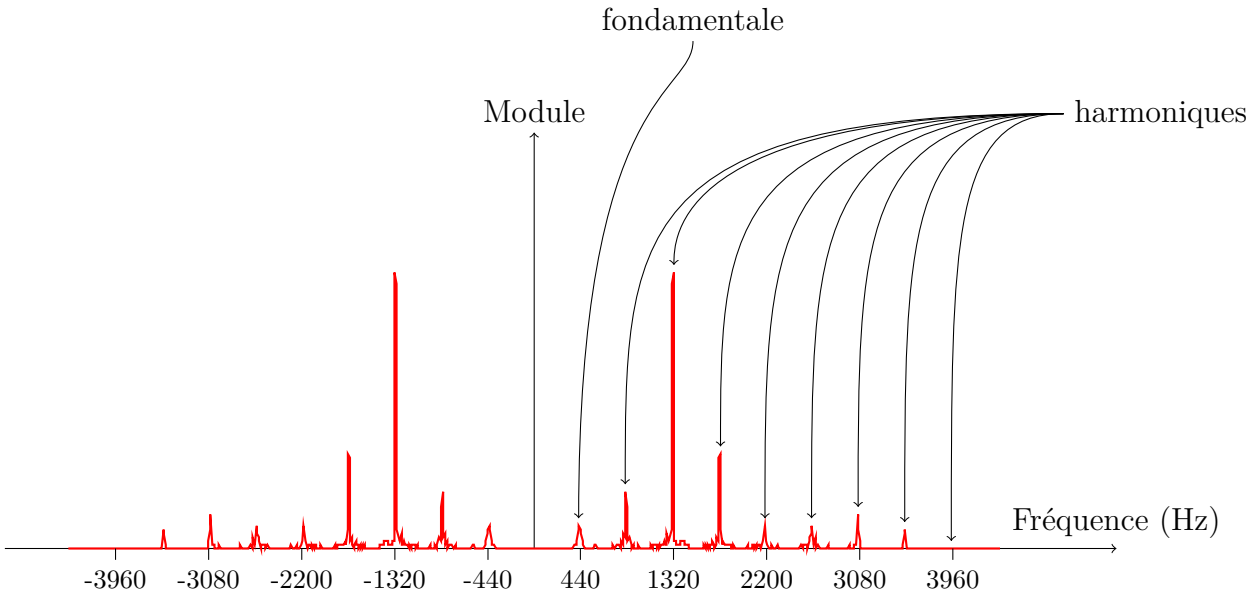

FIGURE 6 – Spectre en amplitude d'un instrument de musique dont la fondamentale est pratiquement absente (hautbois)

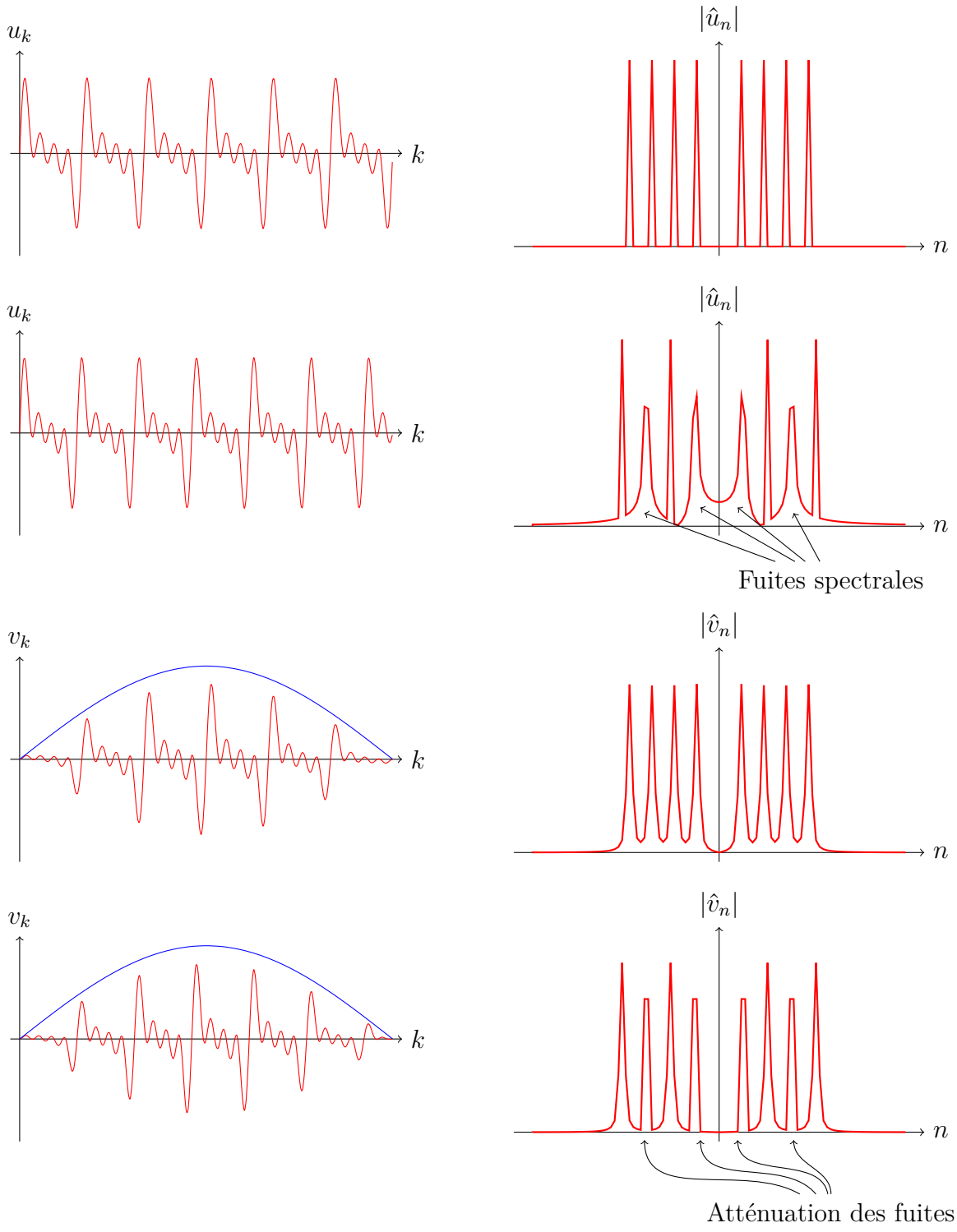

FIGURE 7 – Illustration de l'effet de fuite spectrale sur un signal u à quatre harmoniques. En haut, on a échantillonné le signal sur 6 périodes, puis sur 6.5 périodes. En bas on a multiplié le signal par une fonction de fenêtrage (dessinée en bleu) qui améliore la localisation fréquentielle des pics d'harmoniques dans le spectre lorsqu'on échantillonne sur 6.5 périodes.