Epreuve d'Informatique et Modélisation de Systèmes Physiques

Durée 4 h

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, d'une part il le signale au chef de salle, d'autre part il le signale sur sa copie et poursuit sa composition en indiquant les raisons des initiatives qu'il est amené à prendre.

L'usage de calculatrices est interdit.

La **présentation**, la lisibilité, l'orthographe, la qualité de la **rédaction, la clarté et la précision** des raisonnements entreront pour une **part importante** dans **l'appréciation des copies**. En particulier, les résultats non justifiés ne seront pas pris en compte. Les candidats sont invités à encadrer les résultats de leurs calculs.

Attention : annexe disponible page 19 et 20.

Présentation de la problématique

LA REVERBERATION A CONVOLUTION

A/ La Réverbération

La réverbération peut se définir comme la persistance d'un son après qu'il ait été émis. Après émission d'une onde sonore dans un milieu, cette onde peut être réfléchie en de nombreux points situés à des distances différentes de la source et du récepteur comme l'illustre la figure 1. Le récepteur va donc « entendre » la source mais également la superposition des nombreuses réflexions directes et indirectes qui seront perçues avec un retard temporel par rapport à l'onde captée directement. L'effet de persistance va progressivement s'atténuer. La réverbération joue un rôle très important dans la perception de la musique et de ses harmonies. Une œuvre musicale jouée dans une cathédrale ou dans un opéra ne peut pas être perçue avec la même richesse harmonique sans la réverbération que produisent ces milieux acoustiques particuliers.

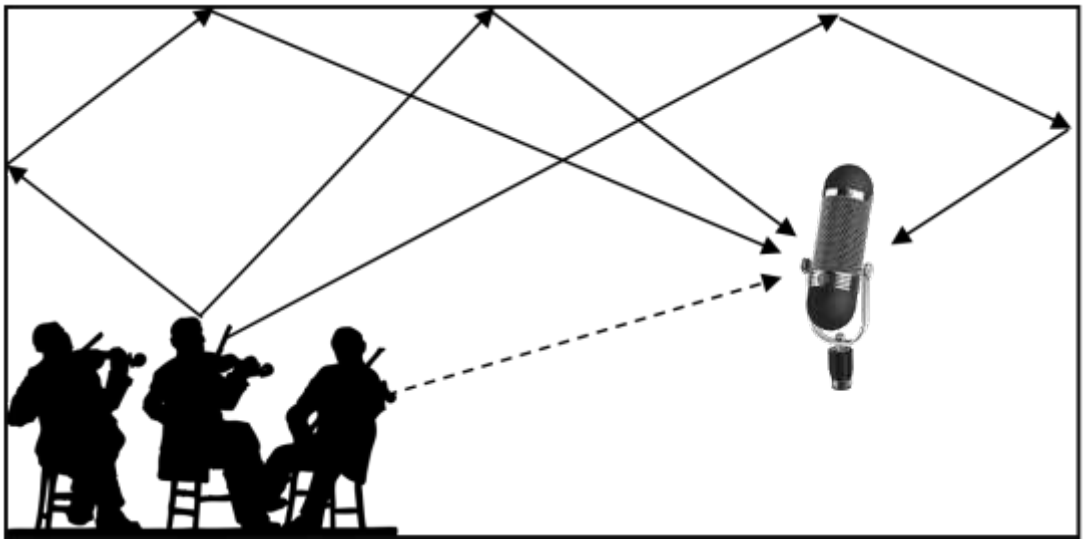


Figure 1 : réverbération

Réverbération numérique

Le rôle de la réverbération numérique est de reproduire l'effet de réverbération sur un signal acoustique enregistré dans des conditions ne permettant pas de produire naturellement un effet de réverbération. C'est très souvent le cas de la musique enregistrée où le récepteur (microphone) est placé très près de la source dans un milieu produisant lui-même très peu de réverbération.

Il existe deux types de réverbérations numériques :

- la réverbération algorithmique qui vient superposer au signal d'origine une multitude de signaux avec des délais (échos) différents et des techniques de filtrage et de combinaisons de ces signaux très complexes,
- la réverbération à convolution qui convolue la réponse impulsionnelle d'un milieu acoustique avec le signal à traiter, permettant ainsi de simuler la réponse d'un milieu

acoustique sans y avoir enregistré le signal pour autant. Nous nous intéresserons dans ce sujet à ce type de réverbération.

Principe de la réverbération à convolution.

Le schéma de la figure 2 ci-dessous illustre le principe de la réverbération à convolution.

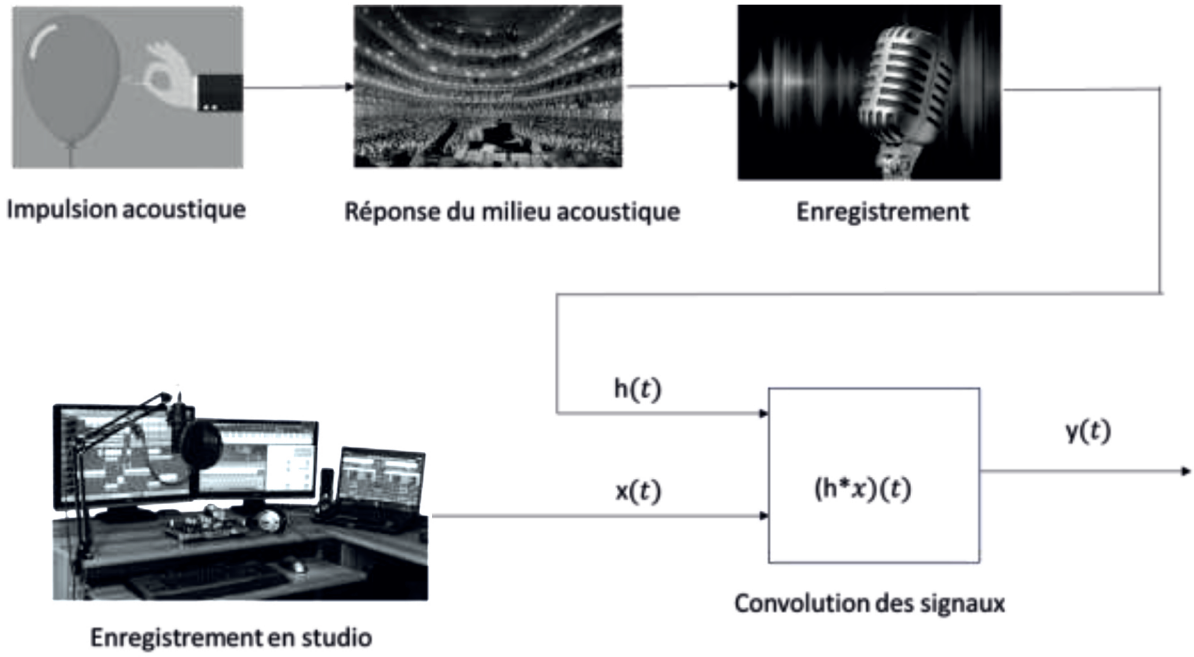


Figure 2 : réverbération à convolution

Afin de réaliser une réverbération à convolution modélisant la réponse d'un milieu acoustique, il est nécessaire d'enregistrer la réponse impulsionnelle de ce milieu. Le microphone utilisé pour cet enregistrement doit permettre de capter le son avec une grande sensibilité pour enregistrer les ondes réfléchies, notamment les plus faibles en amplitude. Le microphone doit également pouvoir capter les ondes sonores selon toutes les directions. Ces deux critères nous conduisent à choisir un microphone électroacoustique plutôt qu'un microphone électrodynamique.

Nous allons dans la suite de ce sujet nous intéresser au fonctionnement du microphone électroacoustique permettant l'enregistrement de la réponse acoustique d'un milieu (Partie I). Nous mettrons ensuite en œuvre les algorithmes permettant de réaliser une réverbération à convolution à partir de la réponse enregistrée (Partie II).

Tournez la page S.V.P.

PREMIERE PARTIE – MODELISATION

MICROPHONE

Un microphone est un transducteur électroacoustique, c'est-à-dire un appareil capable de convertir un signal acoustique en signal électrique.

Parmi les différentes technologies utilisées, deux sont les plus fréquentes :

- les microphones électrodynamiques, qui exploitent le courant induit créé par le déplacement d'une bobine (liée à la membrane) dans un champ magnétique,
- les microphones électroacoustiques, pour lesquels l'onde de pression fait vibrer une membrane, qui constitue l'armature d'un condensateur.

Pour modéliser un microphone électroacoustique, nous nous intéressons pour commencer à un condensateur plan, dont une des armatures est mobile. Nous étudierons ensuite l'intégration de ce condensateur dans un circuit RLC, avant d'observer la réponse de ce circuit en régime sinusoïdal forcé.

LE CONDENSATEUR PLAN (20%)

Un condensateur plan est formé de deux armatures, modélisées par deux plans uniformément chargés, séparés par une distance e.

Une plaque dans le plan (Oyz)*, uniformément chargée avec une densité surfacique* σ*, crée au point* $M(x,y,z)$ *un champ électrostatique* $\vec{E}(M)$ *tel que :*

$$\begin{cases} pour\ x<0, \vec{E}(M) = -\dfrac{\sigma}{2\varepsilon_0}\vec{u}_x \\ pour\ x>0, \vec{E}(M) = +\dfrac{\sigma}{2\varepsilon_0}\vec{u}_x \end{cases}$$

Les effets de bord sont négligés.

Q1. Déterminer et présenter sous forme d'un schéma le champ électrostatique créé, en tout point de l'espace, par un condensateur plan constitué de deux plaques placées en $x = 0$ (charge surfacique $-\sigma$) et en $x = e$ (charge surfacique $+\sigma$)

Q2. En déduire le potentiel $V(x)$ en tout point de l'espace, et représenter graphiquement ce potentiel (on considère l'origine des potentiels : $V(x = 0) = 0$).

La surface des plaques du condensateur plan est notée S et les charges électriques portées par les plaques sont respectivement $-Q$ (plaque en $x = 0$) et $+Q$ (plaque en $x = e$).

Q3. Calculer la tension U entre les deux plaques, en fonction de σ, e et ε_0.

Q4. Déterminer la capacité du condensateur plan, en fonction de e, S et ε_0.

Q5. Calculer l'ordre de grandeur de la capacité d'un condensateur utilisé dans un microphone électrostatique, avec $e \simeq 10\ \mu m$, $S \simeq 1\ cm^2$ et $\varepsilon_0 \simeq 10^{-11}\ USI$.

Q6. Exprimer la densité volumique w_e d'énergie électrique dans le condensateur, puis en déduire l'énergie électrique E_e emmagasinée dans un condensateur soumis à une tension U en fonction de e, S, ε_0 et Q.

Une force $\vec{f} = f\vec{u}_x$ est exercée sur la plaque en $x = e$, ce qui entraîne un déplacement (sans variation d'énergie cinétique) : l'écartement passe de e à $e + \Delta x$.

Q7. Déterminer la variation d'énergie électrique emmagasinée dans le condensateur, sachant que la charge des armatures n'est pas modifiée.

Q8. On peut considérer que cette variation d'énergie correspond au travail de la force $\vec{f}$, déterminer la norme de cette force, en fonction de ε_0, Q et S.

Q9. En déduire la force $\vec{f}_{él}$ exercée par l'armature en $x = 0$ sur celle en $x = e$, qui permet de maintenir celle-ci à l'équilibre.

CIRCUIT ELECTRIQUE (10%)

Le condensateur étudié est désormais utilisé dans un microphone. L'armature placée en $x = 0$ (charge $-Q$) est fixe, alors que celle en $x = e$ (charge $+Q$) peut subir un déplacement Δx, avec $\Delta x \ll e$, en raison de la surpression $p(t)$ due à une onde acoustique.

Plus précisément,
- *en l'absence d'onde acoustique : la pression est P_{atm} et l'armature est en $x = e$;*
- *avec une onde : $P = P_{atm} + p$, avec $p \ll P_{atm}$ et l'armature est en $x = e + \Delta x$.*

Ce déplacement de l'armature induit une charge supplémentaire. Ainsi : $Q = Q_0 + q$, avec $q \ll Q_0$, où Q_0 est la charge statique (pour $x = e$ et $P = P_{atm}$).

Q10. Montrer que la force $\vec{f}_{él}$ de la question précédente peut s'écrire : $\vec{f}_{él}(t) = \vec{f}_{él_0}\left(1 + 2\frac{q}{Q_0}\right)$, où $\vec{f}_{él_0}$ est la force électrique en l'absence de perturbation, soit $Q = Q_0$.

Tournez la page S.V.P.

Le condensateur est un des composants d'un circuit RLC série, alimenté par une source de tension de force électromotrice U_0 et parcouru par un courant d'intensité i.

Q11. Exprimer la tension u_C aux bornes du condensateur, en fonction de Q_0, q, C_0, Δx et e, où C_0 est la capacité du condensateur statique, et la simplifier en ne conservant que les termes d'ordre 1 en $\frac{q}{Q_0}$ et $\frac{\Delta x}{e}$.

Q12. Etablir l'équation électrique du circuit RLC série, pour lier R, L, C_0, U_0, Q_0, e, Δx, q, $\frac{dq}{dt}$ et $\frac{d^2q}{dt^2}$ et la simplifier sachant qu'on choisit la force électromotrice telle que $U_0 = \frac{Q_0}{C_0}$.

La composante statique $\vec{f}_{\text{él}_0}$ est compensée par un dispositif, on ne considère dans la suite que la composante variable : $2\frac{q}{Q_0}\vec{f}_{\text{él}_0}$.

L'armature est aussi soumise à la force de pression $\vec{f}_{press}$, à une force de rappel élastique $\vec{T} = -k\Delta x\vec{u}_x$ et à une force d'amortissement $\vec{f}_{am} = -\lambda\frac{d(\Delta x)}{dt}\vec{u}_x$.

Q13. Exprimer la force $\vec{f}_{press}$ qui s'exerce sur l'armature mobile, en considérant que la pression est P_{atm} dans le condensateur, et P à l'extérieur.

Q14. Ecrire l'équation mécanique qui régit l'évolution de la position de l'armature mobile, de masse m.

REPONSE AU REGIME SINUSOÏDAL FORCE (10%)

On se place en régime sinusoïdal forcé de pulsation ω pour étudier la réponse du microphone à une onde acoustique. p, q et Δx sont des fonctions sinusoïdales du temps à la pulsation ω. On notera $\underline{s}$ la grandeur complexe associée au signal s.

Q15. Justifier l'intérêt d'étudier la réponse à une excitation sinusoïdale.

Q16. Réécrire l'équation électrique et l'équation mécanique en notation complexe, pour obtenir deux équations linéaires qui lient $\underline{q}$, $\underline{\Delta x}$ et $\underline{p}$.

Q17. En déduire une équation entre $\underline{q}$ et $\underline{p}$, sous la forme $\underline{A}\underline{q} = \frac{\underline{p}}{\underline{Z}}$, où $\underline{A} = \frac{j\omega C_0 e}{Q_0 S}(-m\omega^2 + j\lambda\omega + k)$

L'impédance $\underline{Z}$ peut s'écrire $\underline{Z} = \underline{Z}_0 + \underline{Z}_m$, où $\underline{Z}_0 = R + j\left(L\omega - \frac{1}{C_0\omega}\right)$ est l'impédance statique et $\underline{Z}_m$ l'impédance motionnelle.

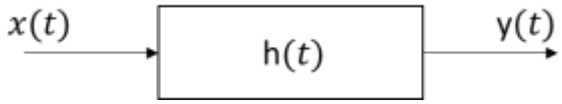
Q18. Montrer que l'impédance motionnelle $\underline{Z}_m = \frac{Q_0^2}{C_0^2 e^2 \omega^2}\frac{1}{\left(jm\omega + \lambda + \frac{k}{j\omega}\right)}$ correspond à l'impédance d'un dipôle $R_m L_m C_m$ en parallèle, et exprimer R_m, L_m et C_m en fonction de m, k, λ, e ,C_0, Q_0 et ω.

Q19. Comment peut-on faire pour « visualiser » à l'oscilloscope l'intensité $i(t)$, image de la surpression $p(t)$ due à l'onde acoustique ?

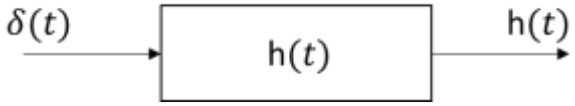
DEUXIEME PARTIE – INFORMATIQUE

ALGORITHME DE CONVOLUTION APPLIQUE A LA REVERBERATION

La réverbération à convolution (voir figure 2, présentation de la problématique) est une adaptation au signal audio du principe des filtres à convolution déjà très utilisés en traitement du signal. On considère que la réverbération d'un milieu est la réponse de ce milieu à un signal sonore. Le milieu acoustique est alors vu comme un filtre linéaire qui applique un traitement au signal sonore de départ :



Dans l' hypothèse des systèmes linéaires invariants, on peut modéliser le filtre par une fonction $h(t)$ qui à toute excitation sonore (signal d'entrée) $x(t)$ fait correspondre une réponse (signal de sortie) $y(t)$. On peut montrer que si le signal d'entrée est une impulsion de Dirac $\delta(t)$ alors la réponse est $y(t) = h(t)$.



Il est possible d'obtenir la réponse impulsionnelle $h(t)$ d'un milieu acoustique quelconque en enregistrant sa réponse à un signal sonore très bref. On pourra alors en déduire la réponse globale de ce même milieu à tout autre signal $x(t)$ par le produit de convolution suivant :

$$y(t) = \int_{-\infty}^{\infty} x(\tau).h(t-\tau).d\tau, \text{ noté également : } (x * h)\,(t). \quad (1)$$

A/ Produit de convolution : réalisation numérique. (15%)

A.1. Signaux numériques à traiter.

On considère dans un premier temps qu'après numérisation les signaux acoustiques enregistrés sont stockés dans deux listes x et h de même dimension n. On notera $x_i = x(t_i)$ et $h_i = h(t_i)$ les valeurs des amplitudes acoustiques des signaux mesurées aux instants t_i.

On notera $dt = t_{i+1} - t_i$ le pas de temps supposé constant et $fe = 1/dt$ la fréquence d'échantillonnage. On considèrera l'instant initial $t_0 = 0\ s$ et l'instant final $T = t_{n-1}$.

Q1. Donner une relation liant les paramètres T, dt et n.

Q2. Ecrire la suite d'instructions permettant de créer une liste tps contenant les valeurs des instants t_i pour un jeu de paramètres : $T = 10\ s$ et $fe = 100\ Hz$.

Afin de tester les algorithmes que nous allons mettre en place, nous allons créer deux listes de signaux virtuels comme définis ci-dessous :

$$h(t) = \cos\left(2.\pi.f_y.t\right).\exp(-t), \text{ avec } f_y = 12\ Hz. \quad (2)$$

$$x(t) = \sum_{k=0}^{3} a_k.\cos(2.\pi.f_{xk}.t + k.\pi), \text{ avec } a \in [1, 0.5, 0.2, 0.1] \text{ et } f_x \in [0.5, 5, 10, 20]\ Hz. \quad (3)$$

Tournez la page S.V.P.

On supposera que le module *math* permettant de définir les fonctions $\cos()$ et $\exp()$ est importé pour les questions suivantes.

Q3. Ecrire une fonction $sigh(t)$ retournant la liste des valeurs $h_i = h(t_i)$ comme définie ci-dessus équation (2). L'argument t sera la liste des valeurs des instants t_i.

Q4. Ecrire une fonction $sigx(t)$ retournant la liste des valeurs $x_i = x(t_i)$ comme définie ci-dessus équation (3). L'argument t sera la liste des valeurs des instants t_i.

La figure 3 donne le tracé des deux fonctions pour le jeu de paramètres défini précédemment.

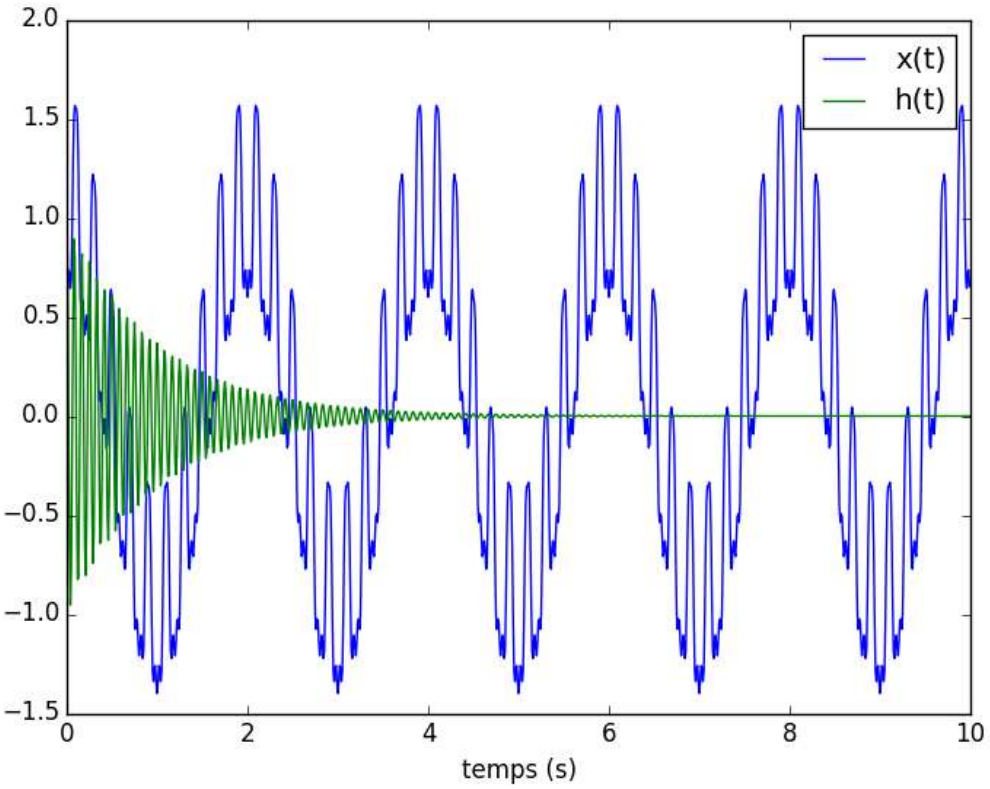


Figure 3 : Signaux tests

L'annexe **E** en fin d'énoncé donne la documentation de la bibliothèque *matplotlib.pyplot* pour la question suivante.

Q5. Ecrire la suite d'instructions permettant d'obtenir le tracé de la figure 3 (légendes et couleurs comprises) à partir des fonctions $sigx(t)$, correspondant à x(t) en bleu sur le tracé réel, et $sigh(t)$ correspondant à h(t) en vert sur le tracé réel et de la liste *tps* créée question Q2.

A.2. Convolution directe.

La première méthode numérique de calcul d'un produit de convolution exploite directement l'expression mathématique (1). Nous allons alors déterminer aux instants t_i les valeurs du signal $y_i = y(t_i)$:

$$y_i = \int_0^{t_{n-1}} x(\tau).h(t_i - \tau).d\tau \quad (4)$$

L'intégrale pour chaque y_i sera évaluée par une méthode des rectangles à gauche appliquée à l'équation (4).

Q6. Ecrire une fonction $convolution(x, h, t)$ qui retourne la liste $y_i = y(t_i)$ des valeurs du résultat du produit de convolution des arguments (listes de valeurs) x et h. L'argument t est la liste des instants t_i correspondants. On admettra que les signaux sont pseudopériodiques. On rappelle que le type de variable liste supporte les indices négatifs tel que pour une liste L de dimension n : $L[n - i] = L[-i]$.

Q7. Donner la complexité d'un appel de la fonction $convolution(x, h, t)$ en fonction de la dimension n des données.

On donne figure 4 ci-dessous le tracé du résultat du produit de convolution obtenu avec les fonctions établies équations (2) et (3).

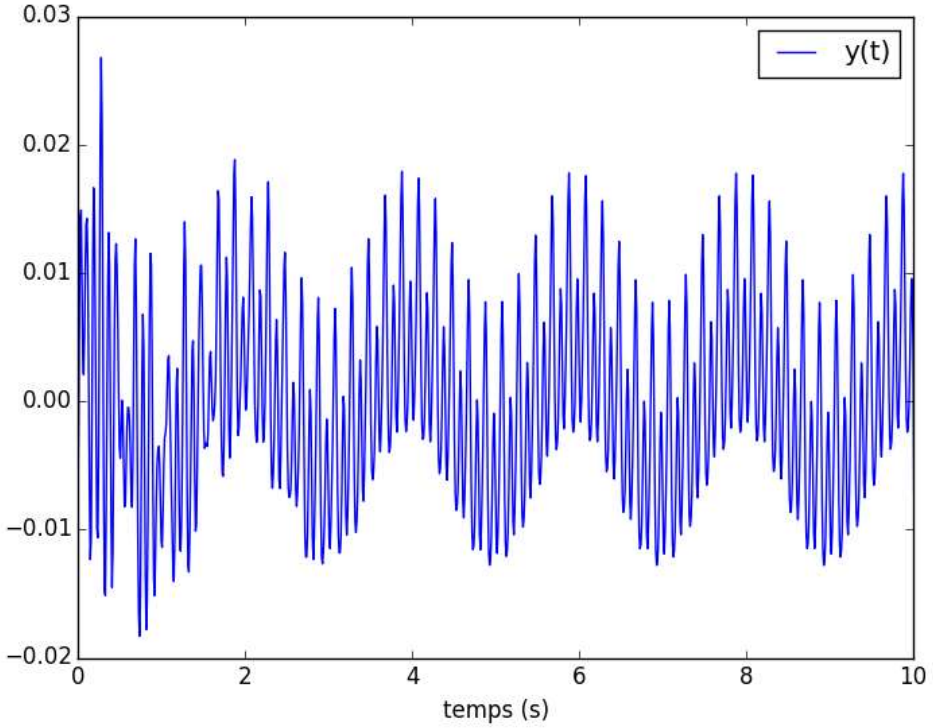


Figure 4 : Résultat du produit de convolution des signaux figure 3

B/ Convolution par transformée de Fourier discrète. (15%)

B.1. Transformée de Fourier Discrète (TFD).

La transformée de Fourier d'un signal temporel permet sa représentation dans une base fréquentielle. Nous allons voir qu'elle peut être employée pour réaliser un produit de convolution.

La fonction $x(t)$ est une fonction du temps t, la fonction transformée $X(f)$ est une fonction de la fréquence f.

On définit la transformée de Fourier $X(f)$ d'une fonction $x(t)$ par (j : unité imaginaire) :

$$X(f) = \int_{-\infty}^{\infty} x(t).\exp(-j.2.\pi.f.t)\,.dt \quad (5)$$

Et la transformée inverse par :

$$x(t) = \int_{-\infty}^{\infty} X(f).\exp(j.2.\pi.f.t).df \quad (6)$$

La figure 5 ci-dessous représente le tracé du module de la transformée de Fourier $X(f)$ du signal $x(t)$ défini équation (3). On utilise usuellement l'expression « spectre fréquentiel » pour ce type de tracé.

Q8. Proposer une interprétation du graphe figure 5 et justifier l'expression de « spectre fréquentiel ».

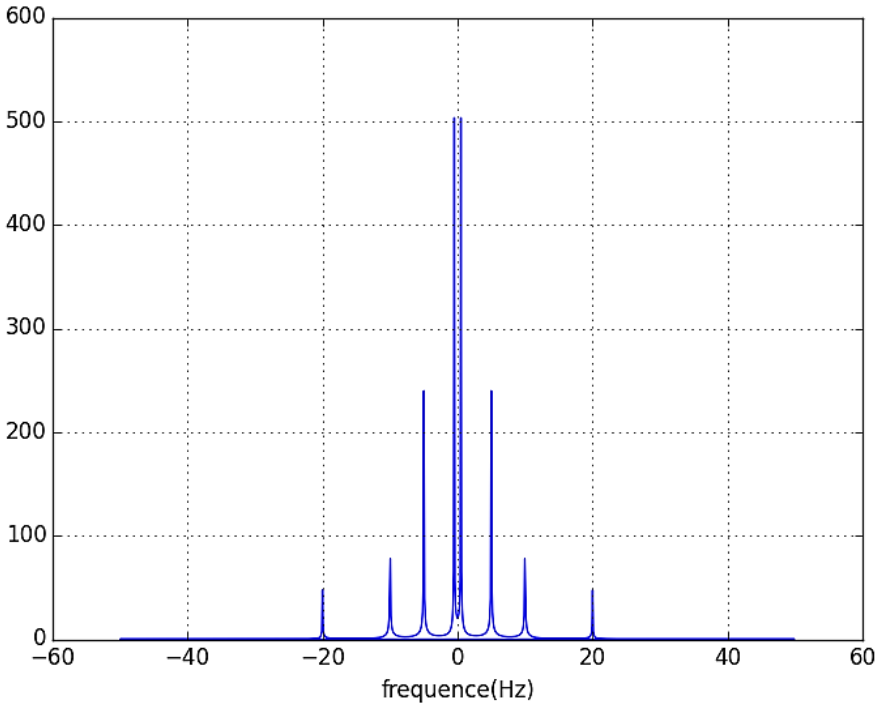


Figure 5 : Transformée de Fourier de x(t) éq (3)

La transformée de Fourier discrète permet une représentation discrète dans une base fréquentielle d'un signal temporel discret. On dispose d'une liste de valeurs discrètes $x_l = x(t_l)$ avec un pas de temps $dt = t_{l+1} - t_l$ supposé constant. La fréquence d'échantillonnage est donc constante et vaut $fe = 1/dt$. Le pas de fréquence vaut alors $df = 1/T$.

On notera $X_k = X(f_k)$ les valeurs de la transformée de Fourier discrète aux fréquences f_k avec :

$$f_k = k.f_e/n \text{ pour } k \in [0, n-1] \quad (7)$$

On notera :

$$\omega_n = \exp(-j.2.\pi/n) \quad (8)$$

Pour calculer la transformée de Fourrier discrète on redéfinit les bornes de l'intégrale dans l'intervalle d'étude $[0, n.dt]$ comme suit :

$$X(f) = \int_0^{n.dt} x(t).\exp(-j.2.\pi.f.t).dt \quad (9)$$

Le signal $x(t)$ est alors supposé nul en dehors de cet intervalle.

Q9. Montrer que si l'on évalue X_k par une méthode des rectangles à gauche appliquée à (9) on obtient :

$$X_k = \sum_{l=0}^{n-1} x_l . \omega_n^{k.l} . dt \text{ (10)}$$

Les grandeurs df et dt étant constantes au cours du calcul il est possible de les prendre en compte à posteriori, on calculera donc dans la suite du sujet la transformée et la transformée inverse pour $dt = 1$ et $df = 1$ respectivement.

On rappelle pour les questions suivantes que la définition des complexes en python peut se faire de la manière suivante :

```
In [74]: a=2.1+3.21j

In [75]: type(a)
Out[75]: complex

In [76]: exp(a)
Out[76]: (-8.1470703008384042-0.55819042818916542j)

In [77]: exp(2.1+3.21j)
Out[77]: (-8.1470703008384042-0.55819042818916542j)
```

On peut alors réaliser toutes les opérations usuelles associées aux nombres complexes.

Q10. Ecrire une fonction $Fourier\_directe(x)$ qui prend pour argument une liste de valeurs d'un signal x et qui retourne la liste des composantes du vecteur (X_k) de la transformée de Fourier discrète de x (pour $dt = 1$). On utilisera les équations (8) et (10).

Q11. Donner la complexité de la détermination d'une transformée de Fourier discrète en fonction de la dimension n des données.

Q12. En analysant la définition de la transformée de Fourier inverse indiquer quelle modification il suffit d'apporter à la fonction $Fourier\_directe(x)$ pour définir une fonction $Fourier\_directe\_inverse(X)$ qui retourne le vecteur (x_k) de la transformée de Fourier discrète inverse de X.

B.2. Convolution avec la TFD.

Il est possible de montrer que le produit de convolution de deux fonctions peut être réalisé dans le domaine fréquentiel grâce à leur transformée de Fourier :

$$\text{Si } y(t) = (x * h)\ (t) \text{ alors } Y(f) = X(f).H(f)$$

Le produit de deux transformées de Fourier discrète est défini par :

$$Y = X.H \text{ où } Y_k = X_k.H_k. \text{ (11)}$$

On peut alors retrouver $y(t)$ par transformée de Fourier inverse.

Q13. Ecrire une fonction $produit(x, h)$ qui prend pour argument deux listes de valeurs et qui retourne le produit de ces deux listes tel que défini par l'équation (11).

Quelles que soient vos réponses aux questions précédentes, on suppose que les fonctions $Fourier\_directe(x)$, $produit(x, h)$ et $Fourier\_directe\_inverse(X)$ sont définies et utilisables pour la suite de cette partie.

Tournez la page S.V.P.

Q14. Donner la suite d'instructions permettant d'obtenir la liste des valeurs y_i résultat du produit de convolution des deux signaux discrets définis équations (2) et (3). On suppose que les listes de valeurs x et h sont déjà définies.

Q15. Evaluer en justifiant précisément votre réponse la complexité de la détermination d'un produit de convolution de deux signaux en utilisant la transformée de Fourier discrète en fonction de la dimension n des données. Comparer à celle du produit de convolution direct déterminée à la question Q11.

C/ Convolution par transformée de Fourier rapide (FFT). (10%)

L'algorithme de la transformée de Fourier rapide ou Fast Fourier Transform a été établi sous sa forme la plus connue par James Cooley et John Tuckey en 1965 bien qu'il reprenne les principes des travaux établis par Carl Friedrich Gauss dès 1805. En réduisant la complexité en temps de l'algorithme de transformée de Fourier discrète, cet algorithme a permis le développement de nombreuses applications en traitement du signal.

Le principe fondamental de cet algorithme est dans un premier temps de diviser le calcul des termes en fréquences paires et impaires :

$$X_k = \sum_{l=0}^{n-1} x_l . \omega_n^{k.l} = \sum_{l=0}^{n/2-1} x_{2l} . \omega_n^{2.k.l} + \omega_n^{k.} \sum_{l=0}^{n/2-1} x_{2l+1} . \omega_n^{2.k.l} \quad (12)$$

On rappelle : $\omega_n = \exp(-j.2.\pi/n)$

On donne figure 6 ci-après le programme python permettant de déterminer récursivement la transformée de Fourier rapide définie équation (12).

```
def Fourier_rapide(x,N):
    if N==1:
        y=x
    else:
        N2=N//2
        #termes impairs
        x0=x[0:N:2]
        y0=Fourier_rapide(x0,N2)
        #termes pairs
        x1=x[1:N:2]
        y1=Fourier_rapide(x1,N2)
        # reconstruction de y(n1+n2*N1)
        y=zeros(N,dtype=complex)
        w=exp(-2j*pi/N)
        W=1
        for n1 in range(0,N2):
            #calcul de y(n1)
            y[n1]=y0[n1]+y1[n1]*W
            #calcul de y(n1+N/2)
            y[n1+N2]=y0[n1]-y1[n1]*W
            #calcul des puissances successives
            W=W*w
    return y
```

Figure 6 : fonction Fourier_rapide

Q16. Montrer en justifiant précisément votre réponse que l'algorithme conduit par les appels récursifs de la fonction $Fourier\_rapide(x, N)$ termine.

Cet algorithme récursif n'est correct que si la taille des données n est une puissance de 2.

Nous allons donc préalablement à son utilisation chercher le plus grand entier N tel que :

$$N \leq n \text{ et il existe un entier } p \text{ tel que } N = 2^p \quad (13)$$

Q17. Ecrire une fonction $nextpow2(n)$ qui retourne N tel que défini ci-dessus (13).

Q18. Avec $N = 2^p$, évaluer le nombre de multiplications réalisées lors d'un appel de la fonction $Fourier\_rapide(x, N)$. En déduire le nombre de multiplications réalisées en fonction de N. Quelle est alors la complexité de l'algorithme ?

Nous allons à présent comparer les trois algorithmes utilisés pour la convolution. Le graphe figure 7 ci-après présente à l'échelle logarithmique les temps de simulation en secondes pour les trois méthodes de calcul numérique du produit de convolution des fonctions (2) et (3) en fonction de la taille des données N.

Q19. Evaluer graphiquement en justifiant aussi précisément que possible vos calculs les ordres de complexités des trois algorithmes en fonction de la taille des données N. Conclure.

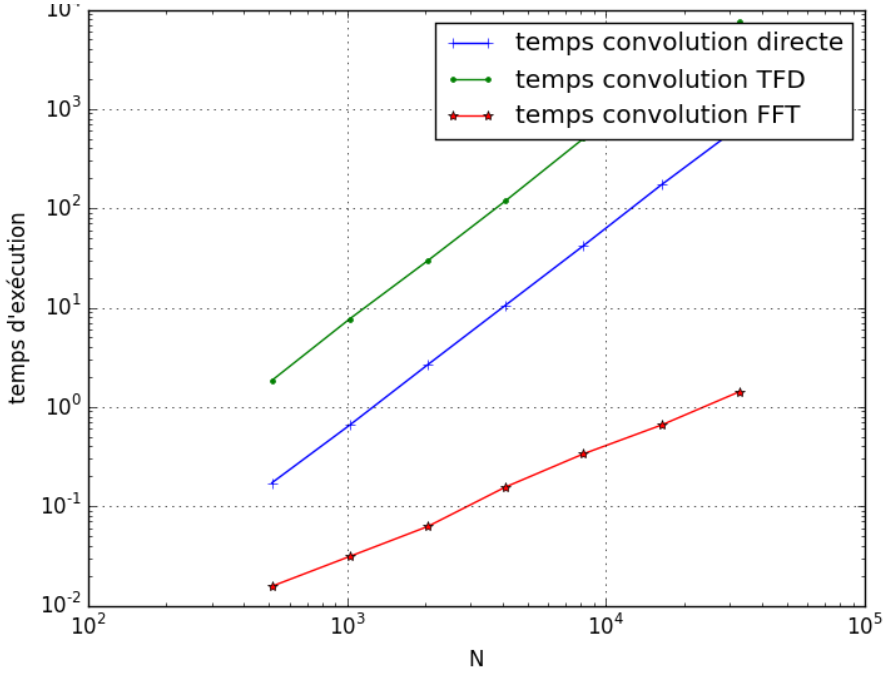


Figure 7 : temps d'exécution des algorithmes de convolution

D/ Application aux fichiers musicaux au format WAVE. (20%)

D.1. Caractéristiques des fichiers audios au format WAVE.

Un fichier WAVE est un fichier binaire dédié au stockage de fichiers audios numériques. Il est constitué d'un premier bloc de 44 octets appelé « en-tête » qui contient les informations relatives au format des données stockées. Le second bloc appelé « data » contient les données elles-mêmes au format WAVE.

Le format WAVE est un format sans compression (contrairement aux fichiers de type mp3 par exemple). Le signal audio est échantillonné à une fréquence fixe et est quantifié pour chaque échantillon. Pour le format WAVE utilisé sur les disques dits CD (disque compact) la fréquence d'échantillonnage est $fech = 44100\ Hz$ et la quantification est réalisée sur un entier signé codé sur 16 bits (2 octets).

Ce couple 44,1kHz-16 bits est un standard couramment employé, c'est celui que nous retiendrons pour la suite. On rappelle que le domaine audible de l'oreille humaine est contenu entre 20 Hz et 20 kHz environ.

Q20. Comment justifier le choix de la fréquence $fech = 44100\ Hz$ pour échantillonner un signal audible ?

Le code utilisé pour la représentation des entiers signés est le complément à 2.

On rappelle le principe du complément à 2 sur n bits :

- L'entier positif est représenté par son code binaire naturel. Le plus grand nombre représentable sur n bits est $2^{n-1} - 1$.
- L'entier négatif $-a$ est représenté par le code binaire naturel de son complément à 2 : $2^n - a$. Le plus petit entier négatif représentable sur n bits est -2^{n-1}.

Q21. Quel est l'intérêt principal du code complément à 2 pour représenter des entiers signés ?

Q22. Quel est le plus grand entier positif représentable sur 16 bits (sans complément à 2) ? En déduire la valeur décimale du plus grand entier positif et la valeur décimale du plus petit entier négatif représentable en complément à 2 sur 16 bits.

On enregistre deux instruments sur un signal stéréo échantillonné à une fréquence d'échantillonnage $fech = 44100$ Hz. Un instrument est enregistré voie gauche, l'autre voie droite.

On donne le tracé figure 8 du fichier obtenu avec le logiciel Audacity dédié au traitement du son. On remarquera que la valeur maximale de l'échantillon correspond à 1 et la valeur minimale à -1 qui sont les amplitudes sonores maximales normées telles qu'affichées par le logiciel.

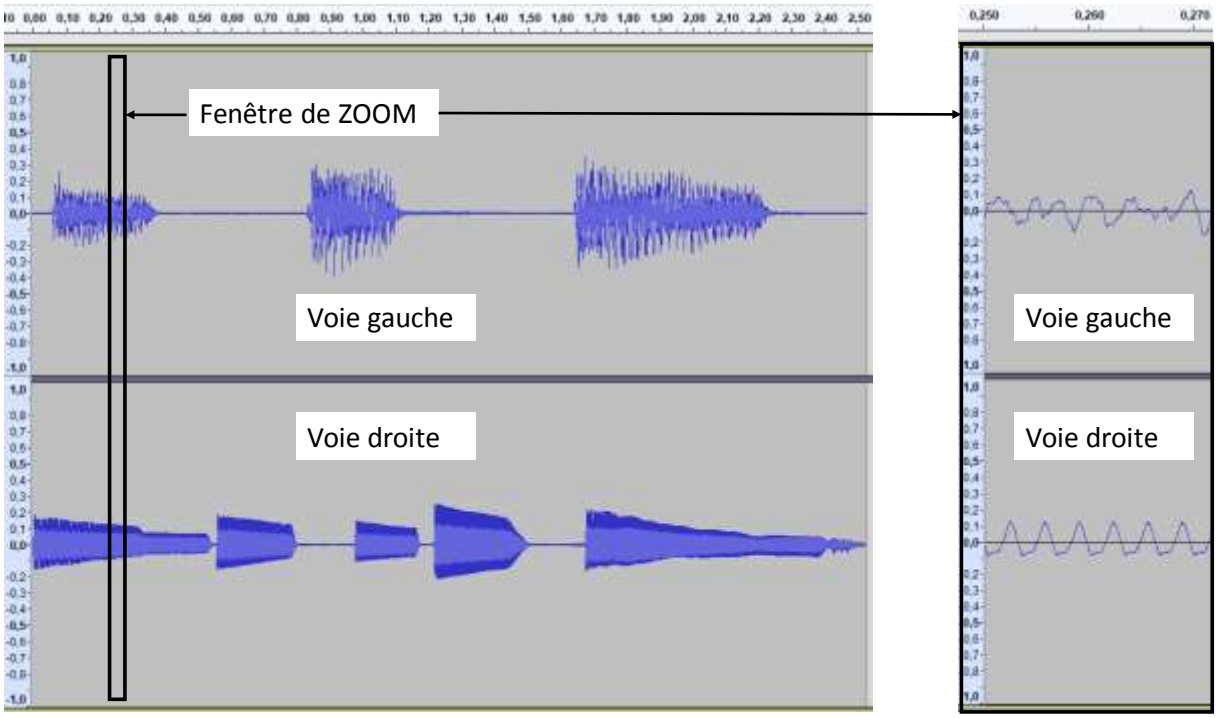


Figure 8 : Signal audio numérique stéréo

Un signal stéréo est constitué de deux voies qui doivent être jouées simultanément. Les données du bloc *data* sont organisées de la manière suivante :

[octets de la voie *gauche* **échantillon 1**] [octets de la voie *droite* **échantillon 1**] [octets de la voie *gauche* **échantillon 2**] [octets de la voie *droite* **échantillon 2**]….. [octets de la voie *gauche* **échantillon n**] [octets de la voie *droite* **échantillon n**].

Chaque échantillon pourra être codé sur plusieurs octets. Les octets sont rangés en ordre « little endian » ce qui signifie que l'octet de poids faible est donné à gauche de l'octet de poids fort.

L'affichage des octets d'un fichier est souvent réalisé au format hexadécimal afin d'en améliorer la lecture, où chaque groupe de 4 bits est représenté par son code hexadécimal. On rappelle ci-dessous la correspondance :

base 10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Base 2	0000	0001	0010	0011	0100	Etc..										
base 16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Ainsi le nombre binaire 234 en base 10 s'écrit en représentation hexadécimale sur 2 octets :

$$(0000\ 0000\ 1110\ 1010)_2 = (00\ \mathrm{EA})_{16}\ big\ endian = (\mathrm{EA}\ 00)_{16}\ little\ endian$$

On peut utiliser pour la programmation en python les fonctions *to_bytes()* et *from_bytes()* pour la conversion des entiers signés en nombres au format binaire sur 2 octets comme illustré ci-après figure 9.

```
In [13]: a=int.to_bytes(234,2,byteorder='little',signed=True)

In [14]: a
Out[14]: b'\xea\x00'

In [15]: type(a)
Out[15]: bytes

In [16]: c=int.from_bytes(b'\xea\x00',byteorder='little',signed=True)

In [17]: c
Out[17]: 234

In [18]: type(c)
Out[18]: int
```

Figure 9 : fonctions to_bytes() et from_bytes

Q23. Que retourne la commande suivante :

int.to_bytes(-32768,2,byteorder='little',signed=True)

Q24. Que retourne la commande suivante :

int.from_bytes(b'\x2a\x00',byteorder='little',signed=True)

Tournez la page S.V.P.

Q25. D'après l'opération réalisée avec les deux variables *a* et *b* de la figure 10 ci-dessous, quel est le nom de l'opération supportée par le type '*bytes'* en python ?

```
In [3]: a=int.to_bytes(12,2,byteorder='little',signed=True)

In [4]: a
Out[4]: b'\x0c\x00'

In [5]: b=int.to_bytes(24,2,byteorder='little',signed=True)

In [6]: b
Out[6]: b'\x18\x00'

In [7]: a+b
Out[7]: b'\x0c\x00\x18\x00'
```

Figure 10 : manipulation des octets

D.2. Convolution de deux fichiers audios au format WAVE.

On donne figure 11 page suivante la fonction *creawave_2oct(voieG,voieD,destination)* qui prend pour argument deux listes *voieG* et *voieD* contenant les valeurs entières des échantillons des voies gauche et droite d'un signal échantillonné à $fech = 44100\ \mathrm{Hz}$ et crée un fichier *destination* au format WAVE sur 2 octets contenant les données des deux voies. Le script python est incomplet et permet d'écrire un fichier WAVE en complétant séquentiellement le fichier (boucle for). On utilise la librairie de fonctions WAVE dont les caractéristiques ne sont pas à connaître.

Q26. Compléter les affectations des variables *g, gbyte*, *d* et *dbyte* de la fonction *creawave_2oct(voieG,voieD,destination)* en définissant les valeurs des échantillons des voies gauches et droites. On normera l'amplitude des signaux : la valeur maximale des listes *voieG* et *voieD* sera codée avec le plus grand entier représentable sur 2 octets. On utilisera les résultats des questions Q23 et Q24.

Q27. Compléter l'argument de la fonction *writeframesraw()* de la fonction *creawave_2oct(voieG,voieD,destination)* : valeur de l'échantillon de deux fois deux octets de type *'bytes'* conformément à la description du format WAVE donnée précédemment. On utilisera le résultat de la questions Q25 notamment.

On souhaite mettre en œuvre les algorithmes établis pour réaliser la convolution des deux fichiers WAVE représentés figures 8 et 12. Le signal figure 8 est le signal à traiter, il est d'une durée de 2.530 secondes. Le signal figure 12 est la réponse impulsionnelle d'un milieu acoustique, il est d'une durée de 1.758 secondes. La figure 13 représente le résultat du produit de convolution tel qu'on souhaite l'obtenir.

Afin d'exploiter pleinement la réverbération, on souhaite que le signal convolué soit d'une durée égale à la somme des durées des deux signaux. Si le signal audio à traiter contient N_1 échantillons et que la réponse impulsionnelle contient N_2 échantillons, Le signal résultat du produit de convolution contiendrait $N = N_1 + N_2 - 1$ échantillons au plus, car l'on doit toujours respecter la condition (13). On calculera donc en pratique (voir Q17) :

$$N = nextpow2(N_1 + N_2 - 1)$$

On complètera donc préalablement les deux signaux à transformer par des zéros.

Q28. Ecrire une fonction *addzeros(voie,N)* qui prend pour arguments un entier *N* et une liste de valeurs numériques *voie* de dimension inférieure à *N* et qui retourne une liste de valeurs de dimension *N* consistant en la liste *voie* complétée par des zéros si *len(voie)* < *N*.

```
def creawave_2oct(voieG,voieD,destination):
    frequence = 44100.0 # hertz
    duree = 1/frequence*len(voieG) # secondes
    wavef = wave.open(destination,'w')# création fichier wave
    wavef.setnchannels(2) # stereo
    wavef.setsampwidth(2) # codage sur 2 octets
    wavef.setframerate(frequence)
    wavef.setnframes(int(duree * frequence)) # nombre d'échantillons
    # Détermination de l'amplitude maximale voie gauche
    MaxiG=max(abs(voieG))
    # Détermination de l'amplitude maximale voie droite
    MaxiD=max(abs(voieD))
    # Détermination de l'amplitude maximale
    Maxi=max(MaxiD,MaxiG)
    for i in range(int(duree * frequence)):
        # valeur échantillon voie gauche
        # valeur échantillon compris dans l'intervalle [-32768,32767]
        g = #A COMPLETER : valeur entière normalisée de l'échantillon
        gbyte= #A COMPLETER : échantillon converti en bytes
        # valeur échantillon voie DROITE
        # valeur échantillon compris dans l'intervalle [-32768,32767]
        d = #A COMPLETER : valeur entière normalisée de l'échantillon
        dbyte= #A COMPLETER : échantillon converti en bytes
        # écriture dans le fichier wave des échantillons
        wavef.writeframesraw(# Argument A COMPLETER)
    wavef.close()
    return None
```

Figure 11 : fonction creawave_2oct

On suppose pour la suite du sujet que les fonctions suivantes créées en amont sont déjà déclarées :

Fonction	Commentaires
$Fourier\_rapide(x, N)$	Retourne la transformée de Fourier discrète d'une liste x de dimension N.
$Fourier\_rapide\_inverse(x, N)$	Retourne la transformée de Fourier discrète inverse d'une liste x de dimension N.
$produit(x, h)$	Retourne une liste résultat du produit de deux transformées de Fourier discrète x et h.
$creawave\_2oct(voieG, voieD, destination)$	Crée un fichier WAVE $destination$ à partir des données entières des échantillons contenus dans les listes $voieG, voieD$.
extrawave_2oct(fichier) Exemple d'appel : voieG, voieD = extrawave_2oct(fichier)	Retourne une séquence de deux listes contenant les valeurs entières des échantillons contenus dans les voies droite et gauche d'un fichier stéréo.
$nextpow2(n)$	Retourne la plus grande puissance de 2 inférieure à n.
$addzeros(voie, N)$	Retourne une liste contenant la liste *voie* complétée par des zéros si *len(voie)* < *N*.

Tournez la page S.V.P.

Q29. Ecrire une fonction $convolution\_reverb(fichier1, fichier2, fichier3)$ qui crée un fichier WAVE fichier3 résultat du produit de convolution des fichiers WAVE fichier1 et fichier2.

Q30. Justifier succinctement que le signal de la figure 13 peut être interprété visuellement comme le signal de la figure 8 perçu avec un effet de réverbération.

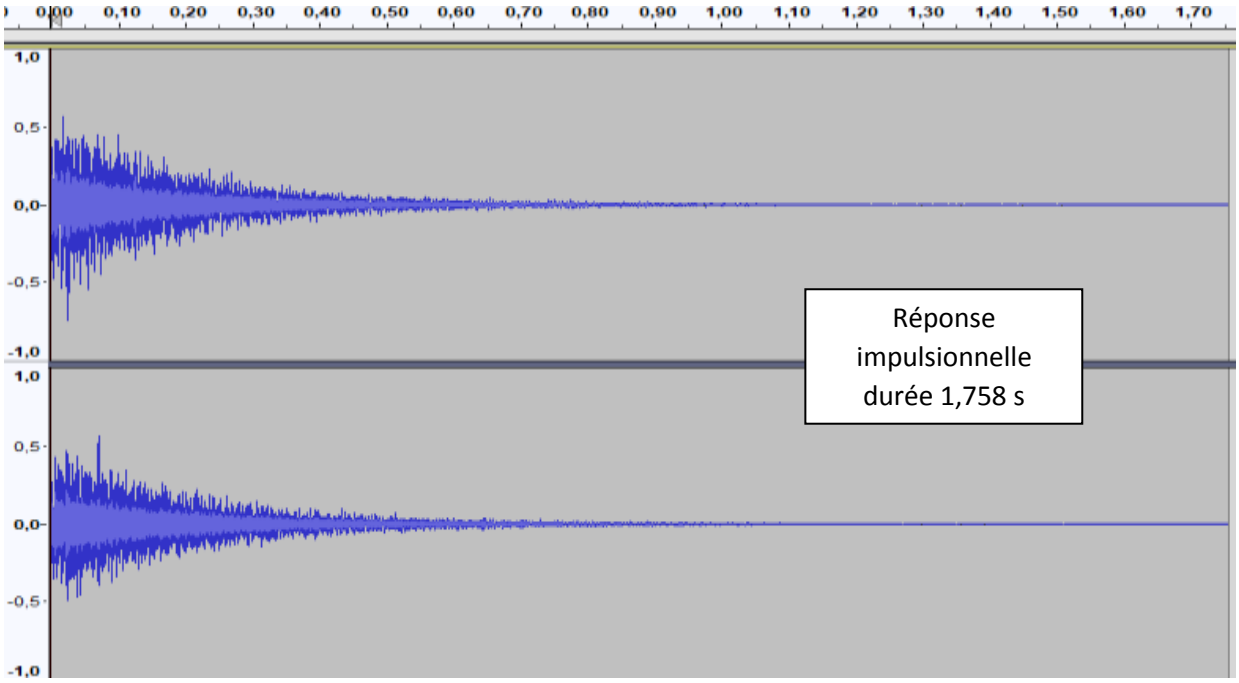


Figure 12 : réponse impulsionnelle

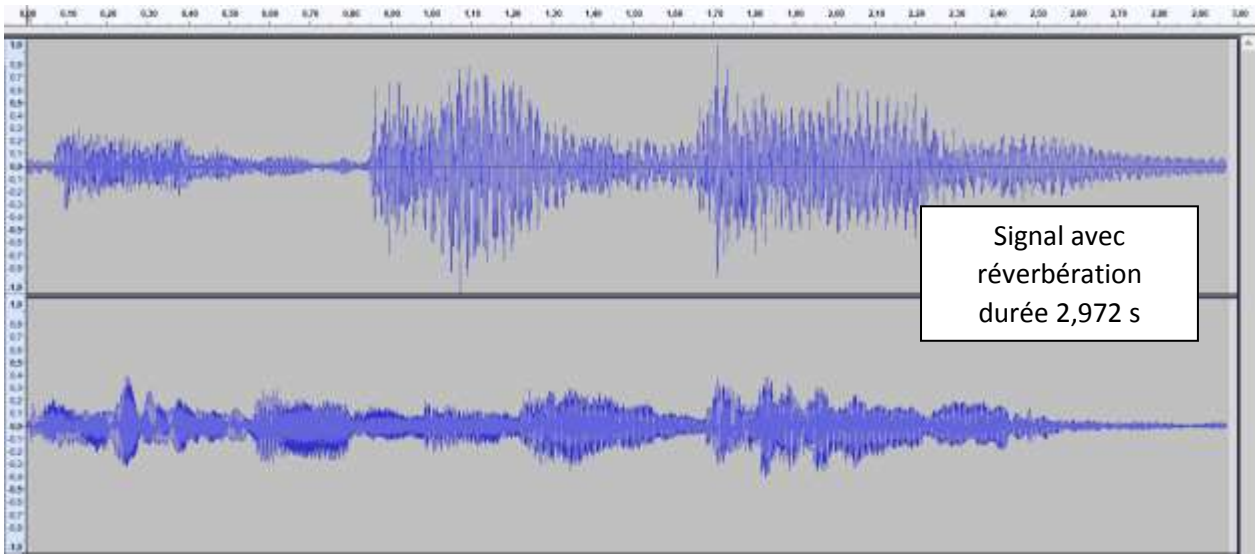


Figure 13 : signal avec réverbération

FIN DE L'ENONCE

E/ Annexe : documentation partielle.

Vous trouverez ci-dessous un extrait de la documentation officielle du module matplotlib.pyplot, dédié au tracé et à la mise en forme de graphiques.

E.1. Fonction plot

`matplotlib.pyplot.plot(*args, **kwargs)`

Plot lines and/or markers to the Axes. *args* is a variable length argument, allowing for multiple *x*, *y* pairs with an optional format string. For example, each of the following is legal:

```
plot(x, y)        # plot x and y using default line style and color
plot(x, y, 'bo')  # plot x and y using blue circle markers
plot(y)           # plot y using x as index array 0..N-1
plot(y, 'r+')     # ditto, but with red plusses
```

An arbitrary number of *x*, *y*, *fmt* groups can be specified, as in:

```
a.plot(x1, y1, 'g^', x2, y2, 'g-')
```

character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'p'	pentagon marker
'*'	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker

Tournez la page S.V.P.

character	description
'd'	thin_diamond marker
'|'	vline marker
'_'	hline marker

The following color abbreviations are supported:

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

E.2. Fonctions annexes pour la mise en forme

En plus de la fonction plot, le module matplotlib.pyplot propose diverses fonctions dédiées à la mise en forme des graphiques. En voici quelques-unes :

- `xlabel(s)` : écrit le contenu de la chaîne s comme étiquette des abscisses.
- `ylabel(s)` : écrit le contenu de la chaîne s comme étiquette des ordonnées.
- `title(s)` : écrit le contenu de la chaîne s comme titre du graphique.
- `legend(L)` : donne une légende au graphique. L doit être une liste de chaînes : L[0] est la légende de la première courbe, L[1] de la deuxième, etc.

IMPRIMERIE NATIONALE – 20 1106 – D'après documents fournis