teaching sciences for innovation

Proposition de corrigé

Concours : Banque PT

Année : 2020

Filière : PT

Épreuve : Informatique et Modélisation

Ceci est une proposition de corrigé des concours de CPGE, réalisée bénévolement par des enseignants de Sciences Industrielles de l'Ingénieur et d'Informatique, membres de l'UPSTI (Union des Professeurs de Sciences et Techniques Industrielles), et publiée sur le site de l'association :

https://www.upsti.fr/espace-etudiants/annales-de-concours

A l'attention des étudiants

Ce document vous apportera des éléments de corrections pour le sujet traité, mais n'est ni un corrigé officiel du concours, ni un corrigé détaillé ou exhaustif de l'épreuve en question.

L'UPSTI ne répondra pas directement aux questions que peuvent soulever ces corrigés : nous vous invitons à vous rapprocher de vos enseignants si vous souhaitez des compléments d'information, et à vous adresser à eux pour nous faire remonter vos éventuelles remarques.

Informez-vous !

Retrouvez plus d'information sur les Sciences de l'Ingénieur, l'orientation, les Grandes Ecoles ainsi que sur les Olympiades de Sciences de l'Ingénieur et sur les Sciences de l'Ingénieur au Féminin sur notre site : www.upsti.fr

L'équipe UPSTI

La réverbération à convolution

Corrigé UPSTI

1 Microphone

1.A Le condensateur plan (20%)

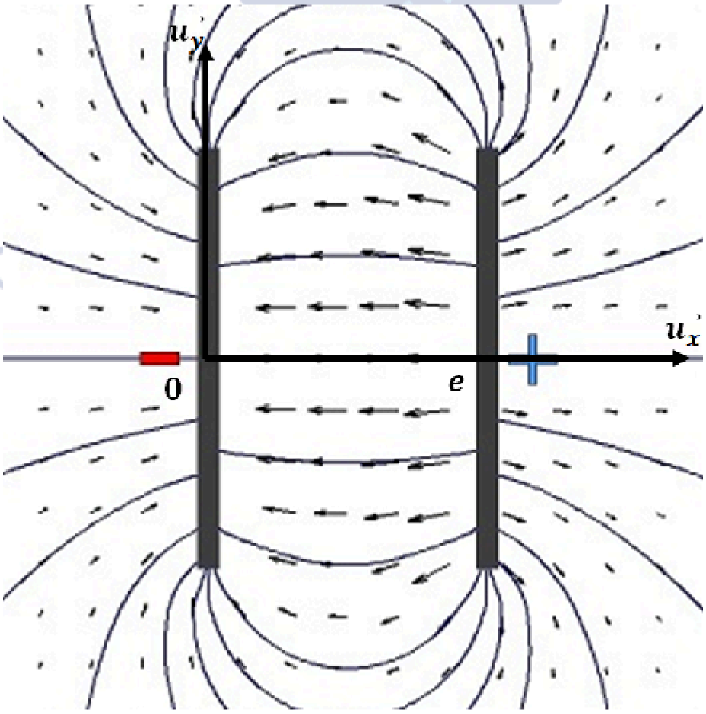
Remarque *Partie modélisation*

Cette première partie porte sur l'étude d'un microphone, et comporte des questions sur :
- Une modélisation du champ électrostatique d'un condensateur plan (20%)
- Une modélisation électrique du circuit RLC (10%)
- L'étude de la réponse au régime sinusoïdal forcé, à l'aide d'impédances complexes (10%)

Ces études et modélisations sont complètement indépendantes de la partie informatique.

Question 1 Déterminer et présenter sous forme d'un schéma le champ électrique 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 $+\sigma$).

L'allure du champ électrique créé par deux plaques (non infinies) distantes de e est :



Si on néglige les effets de bords (ce qui correspond aussi à l'étude de deux plaques parallèles infinies), on voit que le champ électrique est dirigé suivant l'axe $\overrightarrow{u_x}$.

Question 2 En déduire le potentiel $V(x)$ en tout point de l'espace, et représenter graphiquement ce potentiel (on considère l'origine des potentiels : $V(x=0)=0$).

$\overrightarrow{E} = -\overrightarrow{grad}V$

Pour $0 < x < e$, $\left(\underbrace{\frac{-\sigma}{2.\epsilon_0}}_{Plaque1} + \underbrace{\frac{-\sigma}{2.\epsilon_0}}_{Plaque2}\right)\overrightarrow{u_x} = -\frac{\partial V}{\partial x}\overrightarrow{u_x}$

Donc,

$$\boxed{V(x) = \frac{\sigma}{\epsilon_0}.x}$$

Question 3 Calculer la tension U entre les deux plaques, en fonction de σ, e, et ϵ_0.

$$\boxed{U = V(e) = \frac{\sigma}{\epsilon_0}.e}$$

Question 4 Déterminer la capacité du condensateur plan, en fonction de e, S, et ϵ_0.

On sait que $Q = C.U$ (relation électrique) et $\sigma.S = Q$ (par définition de la densité surfacique σ), donc :

$\sigma.S = C.V(e)$

Au final :

$$\boxed{C = \frac{\epsilon_0.S}{e}}$$

Question 5 Calculer l'ordre de grandeur de la capacité d'un condensateur utilisé dans un microphone électrostatique, avec $e \approx 10\mu m$, $S \approx 1cm^2$ et $\epsilon_0 \approx 10^{-11}.USI$

L'application numérique donne :

$$\boxed{C = \frac{10^{-11}.10^{-4}}{10.10^{-6}} = 10^{-8}F = 0.01\mu F}$$

Question 6 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.

$$\boxed{w_e = \frac{1}{2}\epsilon_0.E^2 = \frac{1}{2}\epsilon_0.\left(\frac{\sigma}{\epsilon_0}\right)^2 = \frac{\sigma^2}{2.\epsilon_0}}$$

$$\boxed{E_e = \frac{1}{2}C.U^2 = \frac{1}{2}\epsilon_0.\frac{S}{e}\left(\frac{\sigma.e}{\epsilon_0}\right)^2 = \frac{S.\sigma^2.e}{2.\epsilon_0}}$$

Question 7 Déterminer la variation d'énergie électrique emmagasinée dans le condensateur, sachant que la charge des armatures n'est pas modifiée.

$E_{e-initiale} = \frac{S.\sigma^2.e}{2.\epsilon_0}$

$E_{e-finale} = \frac{S.\sigma^2.(e+\Delta x)}{2.\epsilon_0}$

Donc la variation d'énergie vaut :

$$\boxed{\Delta E_e = E_{e-finale} - E_{e-initiale} = \frac{S.\sigma^2.\Delta x}{2.\epsilon_0}}$$

Question 8 On peut considérer que cette variation d'énergie correspond au travail de la force $\overrightarrow{f}$, déterminer la norme de cette force, en fonction de ϵ_0, Q et S.

$\Delta E_e = W_{\overrightarrow{f}}$

Donc,

$\frac{S.\sigma^2.\Delta x}{2.\epsilon_0} = \overrightarrow{f}.\Delta x.\overrightarrow{u_x}$

La force $\overrightarrow{f}$ est orientée suivant $+\overrightarrow{u_x}$ et sa norme vaut :

$$\boxed{\|\overrightarrow{f}\| = \frac{S.\sigma^2}{2.\epsilon_0} = \frac{Q^2}{2.\epsilon_0.S}}$$

Question 9 En déduire la force $\overrightarrow{f_{él}}$ exercée par l'armature en $x = 0$ sur celle en $x = e$, qui permet de maintenir celle-ci à l'équilibre.

On isole la plaque située en $x = e$. Elle n'est soumise qu'à 2 forces : $\overrightarrow{f}$ et $\overrightarrow{f_{él}}$ (on néglige l'action de pesanteur). Par conséquent, à l'équilibre : $\overrightarrow{f_{él}} = -\overrightarrow{f} = -\frac{Q^2}{2.\epsilon_0.S}.\overrightarrow{u_x}$

1.B Circuit électrique (10%)

Question 10 Montrer que la force $\overrightarrow{f_{él}}$ de la question précédente peut s'écrire : $\overrightarrow{f_{él}} = \overrightarrow{f_{él_0}}\left(1 + 2\frac{q}{Q_0}\right)$ où $\overrightarrow{f_{él_0}}$ est la force électrique en l'absence de perturbation, soit $Q = Q_0$.

D'après les questions 8 et 9, la norme de la force $\overrightarrow{f_{él_0}}$ vaut $\left\|\overrightarrow{f_{él_0}}\right\| = \frac{Q_0^2}{2.\epsilon_0.S}$

De même, on a : $\left\|\overrightarrow{f_{él}}\right\| = \frac{Q^2}{2.\epsilon_0.S} = \frac{(Q_0+q)^2}{2.\epsilon_0.S}$

Comme $q << Q_0$, on trouve :

$\left\|\overrightarrow{f_{él}}\right\| = \frac{Q_0^2+2.Q_0.q+q^2}{2.\epsilon_0.S} \approx \frac{Q_0^2+2.Q_0.q}{2.\epsilon_0.S} \approx \frac{Q_0+2.q}{Q_0}\frac{Q_0^2}{2.\epsilon_0.S}$

Au final,

$$\boxed{\left\|\overrightarrow{f_{él}}\right\| \approx \left(1 + 2\frac{q}{Q_0}\right)\left\|\overrightarrow{f_{él_0}}\right\|}$$

Question 11 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 $\frac{\Delta x}{e}$.

D'après la question 3,

$u_c = \frac{Q_0+q}{S.\epsilon_0}.(e+\Delta x)$

Or

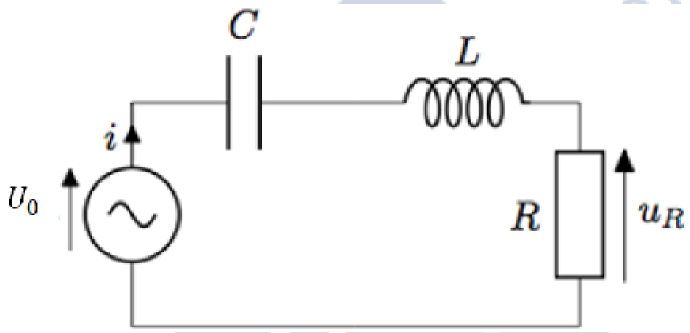
$C_0 = \epsilon_0.\frac{S}{e}$

Donc

$u_c = \frac{Q_0.e+q.e+Q_0.\Delta x+q.\Delta x}{C_0.e} = \frac{Q_0}{C_0}.\left(1+\frac{q}{Q_0}+\frac{\Delta x}{e}+\frac{q.\Delta x}{Q_0.e}\right)$

$$\boxed{u_c \approx \frac{Q_0}{C_0}.\left(1+\frac{q}{Q_0}+\frac{\Delta x}{e}\right)}$$

Question 12 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 $U_0 = \frac{Q_0}{C_0}$.



$U_0 = u_R + u_L + u_C = R.I + L.\frac{di}{dt} + u_C$

Puis en injectant l'expression de la capacité du condensateur C :

$i = C.\frac{du_C}{dt} = \epsilon_0.\frac{S}{e+\Delta x}.\frac{du_C}{dt}$

Donc ,

$U_0 = R.C.\frac{du_C}{dt} + L.C.\frac{d^2u_C}{dt^2} + u_C$

Pour effectuer les dérivées temporelles, on suppose que Δx bouge très peu et donc reste constant. Cette hypothèse n'est pas vraie en réalité mais l'armature du condensateur va bouger/évoluer beaucoup moins vite que la charge q.

$U_0 = \frac{R.C}{C_0}.\frac{dq}{dt} + \frac{L.C}{C_0}.\frac{d^2q}{dt^2} + \frac{Q_0}{C_0}.\left(1+\frac{q}{Q_0}+\frac{\Delta x}{e}\right)$

$\frac{R.C}{C_0}.\frac{dq}{dt} + \frac{L.C}{C_0}.\frac{d^2q}{dt^2} + \frac{q}{C_0} = -\frac{Q_0.\Delta x}{C_0.e}$ car $U_0 = \frac{Q_0}{C_0}$

D'autre part, on a vu que $C_0 = \frac{\epsilon_0.S}{e}$ et donc $C = \frac{\epsilon_0.S}{e+\Delta x}$. On en déduit :

$\frac{C_0}{C} = \frac{e+\Delta x}{e}$

On réinjecte dans l'équation précédente :

$\frac{R.e}{e+\Delta x}.\frac{dq}{dt} + \frac{L.e}{e+\Delta x}.\frac{d^2q}{dt^2} + \frac{q}{C_0} = -\frac{Q_0.\Delta x}{C_0.e}$

$R.\left(1-\frac{\Delta x}{e}\right).\frac{dq}{dt} + L.\left(1-\frac{\Delta x}{e}\right).\frac{d^2q}{dt^2} + \frac{q}{C_0} = -\frac{Q_0.\Delta x}{C_0.e}$

On néglige les termes en $\frac{\Delta x}{e}$ par rapport à 1.

$$\boxed{R.\frac{dq}{dt} + L.\frac{d^2q}{dt^2} + \frac{q}{C_0} = -\frac{Q_0.\Delta x}{C_0.e}}$$

Question 13 Exprimer la force $\overrightarrow{f_{press}}$ qui s'exerce sur l'armature mobile, en considérant que la pression est P_{atm} dans le condensateur, et P à l'extérieur.

$\overrightarrow{f_{press}} = -(P - P_{atm}).S.\overrightarrow{u_x} = -(P_{atm} + p - P_{atm}).S.\overrightarrow{u_x}$

$$\boxed{\overrightarrow{f_{press}} = -p.S.\overrightarrow{u_x}}$$

Question 14 Ecrire l'équation mécanique qui régit l'évolution de la position de l'armature mobile, de masse m.

On isole l'armature mobile, et on écrit le TRD sur $\overrightarrow{u_x}$:

$$\boxed{\overrightarrow{f_{press}} + 2.\frac{q}{Q_0}.\overrightarrow{f_{élo}} - k.\Delta x.\overrightarrow{u_x} - \lambda.\frac{d(\Delta x)}{dt}.\overrightarrow{u_x} = m.\frac{d^2(\Delta x)}{dt^2}.\overrightarrow{u_x}}$$

1.C Réponse au régime sinusoïdal forcé (10%)

Question 15 Justifier l'intérêt d'étudier la réponse à une excitation sinusoïdale.

Le système a pour vocation à reproduire des sons. Nous pourrons ainsi savoir comment il va reproduire chaque fréquence/note.

Question 16 Réécrire l'équation électrique et l'équation mécanique en notation complexe, pour obtenir deux équations linéaires qui lient $\underline{q}$, $\underline{\Delta x}$ et $\underline{p}$.

Equation "électrique" : $\left(R.j.\omega - L.\omega^2 + \frac{1}{C_0}\right)\underline{q} = \frac{-Q_0.\underline{\Delta x}}{C_0.e}$

Equation "mécanique" : $-\underline{p}.S - \frac{Q_0}{\epsilon_0.S}.\underline{q} = (k + \lambda.j.\omega - m.\omega^2).\underline{\Delta x}$

Question 17 En déduire une relation entre $\underline{p}$ et $\underline{q}$, 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.

$-\underline{p}.S - \frac{Q_0}{\epsilon_0.S}.\underline{q} = -(k + \lambda.j.\omega - m.\omega^2).\left(R.j.\omega - L.\omega^2 + \frac{1}{C_0}\right)\underline{q}.\frac{C_0 e}{Q_0}$

$\left[-\frac{Q_0}{\epsilon_0.S} + (k + \lambda.j.\omega - m.\omega^2).\left(R.j.\omega - L.\omega^2 + \frac{1}{C_0}\right)\frac{C_0 e}{Q_0}\right]\underline{q} = \underline{p}.S$

$\left[-\frac{Q_0}{\epsilon_0.S^2} + (k + \lambda.j.\omega - m.\omega^2).\left(R + j.\left(L.\omega - \frac{1}{\omega.C_0}\right)\right)\frac{j\omega C_0 e}{Q_0.S}\right]\underline{q} = \underline{p}$

On reconnait le terme $\underline{A}$ donné dans l'énoncé :

$\left[-\frac{Q_0}{\epsilon_0.S^2} + \underline{A}.\left(R + j.\left(L.\omega - \frac{1}{\omega.C_0}\right)\right)\right]\underline{q} = \underline{p}$

$\left[-\frac{Q_0}{\epsilon_0.S^2.\underline{A}} + \left(R + j.\left(L.\omega - \frac{1}{\omega.C_0}\right)\right)\right]\underline{A}\underline{q} = \underline{p}$

On retrouve l'équation demandée en posant :

$$\boxed{\underline{Z} = -\frac{Q_0}{\epsilon_0.S^2.\underline{A}} + \left(R + j.\left(L.\omega - \frac{1}{\omega.C_0}\right)\right)}$$

Question 18 Montrer que l'impédance motionnelle $\underline{Z_m} = \frac{Q_0^2}{C_0^2 e^2 \omega^2 \left(jm\omega + \lambda + \frac{k}{j\omega}\right)}$ 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.

D'après la question précédente, on identifie l'impédance motionnelle :

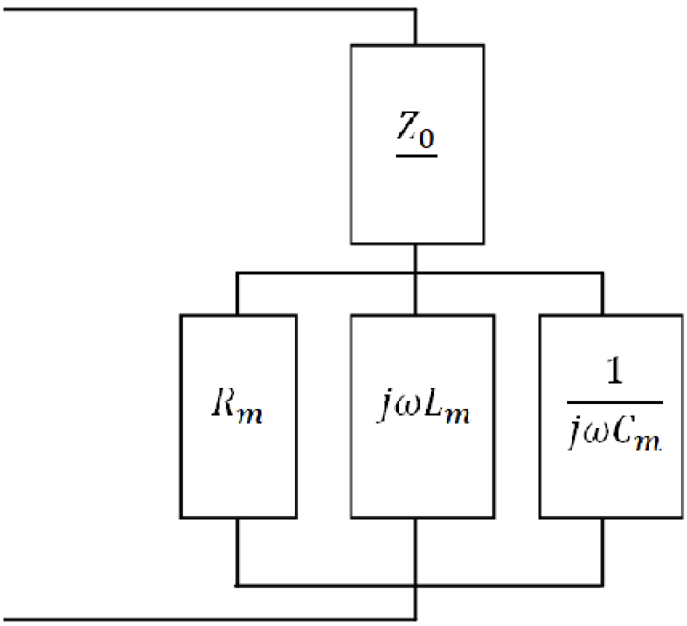
$\underline{Z_m} = -\frac{Q_0}{\epsilon_0.S^2.\underline{A}} = -\frac{Q_0}{\epsilon_0.S^2.\frac{j\omega C_0 e}{Q_0 S}(-m\omega^2 + j\lambda\omega + k)}$

Or $\epsilon_0.S = C_0.e$.

$\underline{Z_m} = -\frac{Q_0^2}{j\omega(C_0 e)^2(-m\omega^2 + j\lambda\omega + k)}$

$$\boxed{\underline{Z_m} = \frac{Q_0^2}{\omega^2 (C_0 e)^2 (jm\omega + \lambda + \frac{k}{j\omega})}}$$

L'impédance $\underline{Z}$ est équivalente au schéma suivant :



$\underline{Z} = \underline{Z_0} + \frac{1}{\frac{1}{R_m} + \frac{1}{L_m.j\omega} + C_m.j\omega}$

Donc,

$\underline{Z_m} = \frac{1}{\frac{1}{R_m} + \frac{1}{L_m.j\omega} + C_m.j\omega} = \frac{L_m}{\frac{L_m}{R_m} + \frac{1}{j\omega} + C_m.L_m.j\omega}$

$\frac{L_m}{\frac{L_m}{R_m} + \frac{1}{j\omega} + C_m.L_m.j\omega} = \frac{Q_0^2}{\omega^2 (C_0 e)^2 (-jm\omega - \lambda - \frac{k}{j\omega})}$

$$\left\{\begin{array}{l} R_m = \frac{Q_0^2}{\omega^2.C_0^2.e^2.\lambda} \\ L_m = \frac{Q_0^2}{\omega^2.C_0^2.e^2.k} \\ C_m = \frac{\omega^2.C_0^2.e^2.m}{Q_0^2} \end{array}\right.$$

Question 19 Comment peut-on faire pour "visualiser" à l'oscilloscope l'intensité $i(t)$, image de la surpression $p(t)$ due à l'onde acoustique ?

Pour visualiser directement le courant $i(t)$, il suffit de mesurer la tension aux bornes de l'impédance $\underline{Z_0}$ ou de la résistance R.

2 Algorithme de convolution appliqué à la réverbération

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

2.A.1 Signaux numériques à traiter

Question 1 Donner une relation liant les paramètres T, dt et n.

$$\boxed{T = (n-1) \times dt}$$

$T = t_{n-1}$, il y a donc $n-1$ intervalles $dt = t_{i+1} - t_i$ entre t_0 et T.

Question 2 Ecrire la suite d'instructions permettant de créer une liste `tps` contenant les valeurs des instants t_i pour un jeu de paramètres : $T = 10\,\mathrm{s}$ et $f_e = 100\,\mathrm{Hz}$.

En écartant l'utilisation de `np.linspace` (qui rend l'exercice trivial et ne semble pas être dans l'esprit de la question), l'utilisation d'une boucle `while` ou d'une boucle `for` sont possibles. Vu la question précédente, la réponse attendue est probablement la première puisqu'elle calcule le nombre d'éléments à ajouter.

```python
# Crée une liste tps contenant les valeurs des instants ti
# à l'aide d'une boucle for et de range : on calcule le nombre de dates à ajouter (n),
# puis on les ajoute à la liste tps
T = 10 # 10 secondes
fe = 100 # 100 Hz

n = T * fe + 1 # Nombre d'instants ti
tps = [] # La liste tps va contenir les valeurs des ti
for i in range(n):
  tps.append(i / fe)
```

Avec une boucle while :

```python
# Crée une liste tps contenant les valeurs des instants ti
# à l'aide d'une boucle while : tant que T n'est pas atteint, on ajoute dt à l'instant précédant
T = 10 # 10 secondes
fe = 100 # 100 Hz

dt = 1 / fe
tps = [0] # La liste tps va contenir les valeurs des ti, à commencer par t0 = 0
while tps[-1] < T:
  tps.append(ti[-1] + dt)
```

Question 3 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.

L'argument est une liste, et il faut appliquer une fonction sur chaque élément de cette liste. On peut définir une fonction intermédiaire :

```python
def calcul_h(ti, fy):
  # Calcule la valeur hi = h(ti)
```

```python
  return math.cos(2*math.pi*fy*ti)*math.exp(-ti)

def sigh(t):
  # Retourne la liste h des valeurs hi = h(ti)
  fy = 12 # fy = 12 Hz

  h = []  # liste qui contiendra les valeurs hi
  for ti in t:
    h.append(calcul_h(ti, fy))

  return h
```

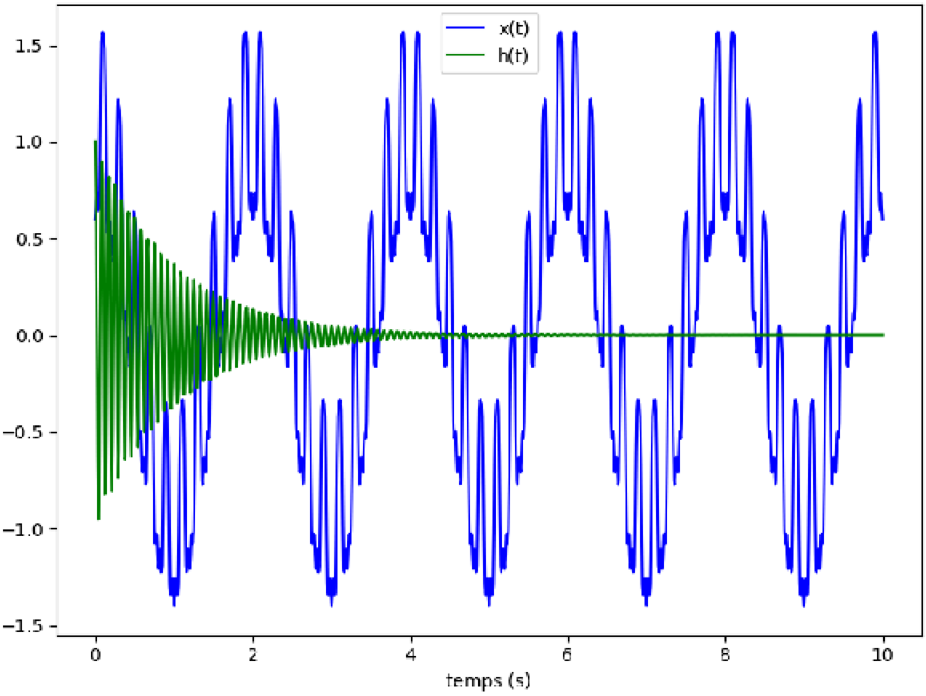
Question 4 Ecrire une fonction `sigx(t)` retournant la liste des valeurs $x_i = x(t_i)$ comme définie ci-dessus équation (3). L'argument `t` sera la liste des valeurs des instants t_i.

L'argument est une liste, et il faut appliquer une fonction sur chaque élément de cette liste. On peut définir une fonction intermédiaire :

```python
def calcul_x(ti, a, fx):
  # Calcule la valeur xi = x(ti) en fonction des valeurs ak et fxk
  x = 0 # accumulateur pour la somme
  for k in range(len(a)): # la valeur de k est nécessaire, on parcourt donc par indice et non par valeurs
    x += a[k]*math.cos(2*math.pi*fx[k]*ti + k*math.pi)

  return x

def sigx(t):
  # Retourne la liste x des valeurs xi = x(ti)
  a = [1, 0.5, 0.2, 0.1] # Sans unité
  fx = [0.5, 5, 10, 20] # en Hz

  x = []  # liste qui contiendra les valeurs xi
  for ti in t:
    x.append(calcul_x(ti, a, fx))

  return x
```

Question 5 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 `sigh(t)` en vert sur le tracé réel et de la liste `tps` créée question Q2.

```python
import matplotlib.pyplot as plt

plt.plot(tps, sigx(tps), "b")
plt.plot(tps, sigh(tps), "g")
plt.xlabel("temps (s)") # Etiquette sur les abscisses, pas d'étiquette sur les ordonnées
plt.legend(["x(t)", "h(t)"])

plt.show()
```



2.A.2 Convolution directe

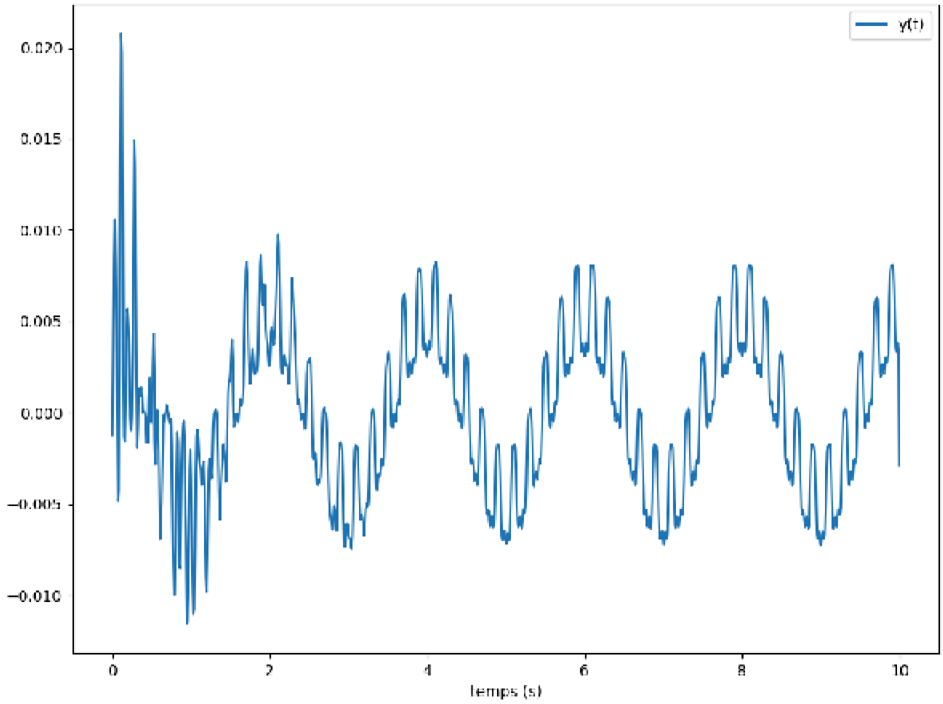
Question 6 Ecrire une fonction `convolution(x,h,t)` qui retourne la liste $y_i = y(ti)$ des valeurs du résultat du produit de convolution des arguments (listes de valeurs) `x` et `h`. L'argument `t` est la liste des instants t_i correspondants. On admettra que les signaux sont pseudopériodiques. On rappelle que le type de variable liste supporte les indices négatifs tel que pour une liste `L` de dimension `n` : `L[n-i] = L[-i]`.

```python
def calcul_y(x, h, t, i):
    # Calcul de yi = y(ti), la valeur de y d'indice i,.
    # x, h et t sont des listes de valeurs et d'instants

    yi = 0 # Accumulateur pour yi

    # Largeur d'un rectangle :
    dt = (t[-1] - t[0]) / (len(t) - 1) # Il y a un rectangle de moins qu'il n'y a d'instants ti
    for k in range(len(t)-1): # Boucle sur les indices et non sur les valeurs de ti
        yi += x[i] * h[i - k] # Accumulation des hauteurs des rectangles à gauche
        # L'indice i - k peut être négatif, mais comme les signaux sont pseudopériodiques on obtiendra une valeur très proche de celle du signal réel avant t0.

    return yi * dt # Somme des hauteurs des rectangles en multipliant par leur largeur

def convolution(x,h,t):
    # Renvoi la liste des yi, en calculant pour chaque élément de t la valeur yi correspondante
    y = []
    for i in range(len(t)):
        y.append(calcul_y(x,h,t,i))
    return y
```

Question 7 Donner la complexité d'un appel de la fonction `convolution(x,h,t)` en fonction de la dimension `n` des données.

Les listes `x`, `h` et `t` sont de longueurs `n`. La fonction `convolution` réalise `n` appels à la fonction `calcul_y(x, h, t, i)`. La fonction `calcul_y(x, h, t, i)` réalise `n` itérations de sa boucle, qui ne contient que des opérations élémentaires.

La complexité d'un appel de la fonction `convolution(x,h,t)` est donc en $\boxed{O(n^2) \text{ opérations élémentaires}}$.



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

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

Question 8 Proposer une interprétation du graphe figure 5 et justifier l'expression de «spectre fréquentiel».

Le signal $x(t)$ est une somme de 4 sinusoïdes de fréquences 0,5, 5, 10 et 20 Hz, d'amplitudes différentes. Sur le graphe, on peut remarquer 4 pics qui correspondent à la décomposition de ce signal en 4 paires de sinusoïdes (4 harmoniques) correspondant à ces fréquences (chaque paire étant formée d'une fréquence positive et d'une fréquence négative de même valeur absolue). L'amplitude des pics correspond aux amplitudes des sinusoïdes (par exemple, en sommant les amplitudes des deux pics de fréquence 20 Hz et −20 Hz, on retrouve bien l'amplitude de $0,1$ utilisée pour générer le signal).

C'est donc bien un spectre fréquentiel du signal, puisqu'il permet de représenter la composition du signal en sinusoïdes de différentes fréquences, comme un spectre d'une onde lumineuse représente les différentes longueurs d'ondes qui la composent.

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

$$X_k = \sum_{l=0}^{n-1} x_l \cdot \omega_n^{k \cdot l} \cdot dt$$

$$\begin{aligned} X_k = X(f_k) &= \int_0^{n \cdot dt} x(t) \cdot \exp(-j \cdot 2 \cdot \pi \cdot f_k \cdot t) \cdot dt \\ &= \sum_{l=0}^{n-1} \int_{l \cdot dt}^{(l+1) \cdot dt} x(t) \cdot \exp(-j \cdot 2 \cdot \pi \cdot f_k \cdot t) \cdot dt \end{aligned}$$

En faisant l'hypothèse que

$$\forall l \in [\![0; n-1]\!], \forall t \in [l \cdot dt; (l+1) \cdot dt], x(t) \cdot \exp(-j \cdot 2 \cdot \pi \cdot f_k \cdot t) \approx x(l \cdot dt) \cdot \exp(-j \cdot 2 \cdot \pi \cdot f_k \cdot l \cdot dt)$$

(méthode des rectangles à gauche), on peut écrire :

$$\begin{aligned} X_k &= \sum_{l=0}^{n-1} \int_{l \cdot dt}^{(l+1) \cdot dt} x(l \cdot dt) \cdot \exp(-j \cdot 2 \cdot \pi \cdot f_k \cdot l \cdot dt) \cdot dt \\ &= \sum_{l=0}^{n-1} dt \cdot x(l \cdot dt) \cdot \exp(-j \cdot 2 \cdot \pi \cdot f_k \cdot l \cdot dt) \end{aligned}$$

On identifie $x(l \cdot dt) = x(t_l) = x_l$, et $f_k \cdot l \cdot dt = (k \cdot f_e/n) \cdot l \cdot dt = k \cdot l/n$, donc :

$$\begin{aligned} X_k &= \sum_{l=0}^{n-1} dt \cdot x_l \cdot \exp(-j \cdot 2 \cdot \pi \cdot k \cdot l/n) \\ &= \sum_{l=0}^{n-1} dt \cdot x_l \cdot \exp(-j \cdot 2 \cdot \pi/n)^{k \cdot l} \\ &= \sum_{l=0}^{n-1} x_l \cdot \omega_n^{k \cdot l} \cdot dt \end{aligned}$$

Question 10 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).

```python
def calcul_Xk(k, x, wn):
  # Calcul du k-ieme terme du vecteur (Xk)
  # par transformée de Fourier discrète
  # x est la liste des xk
  # wn est le terme omega_n

  Xk = 0 # Accumulateur pour le k-ième terme du vecteur (Xk)
  for l in range(len(x)):
    Xk += x[l] * (wn ** (k * l))

  return Xk

def Fourier_directe(x):
  # Calcul du vecteur (Xk)
  # par transformée de Fourier discrète
  n = len(x)
  wn = math.exp(-1j*2*math.pi/n)
```

```python

    Xk = []
    for k in range(n):
      Xk.append(calcul_Xk(k,x,wn))

    return Xk
```

Question 11 Donner la complexité de la détermination d'une transformée de Fourier discrète en fonction de la dimension n des données.

Soit `n` la dimension de `x`, c'est-à-dire le nombre de valeurs de cette liste. La fonction `Fourier_directe` fait `n` appels à la fonction `calcul_Xk`, qui réalise `n` additions de termes dans l'accumulateur. Les autres opérations sont à coût constant.

La complexité asymptotique de la détermination d'une transformée de Fourier discrète est donc en $\boxed{O(n^2)}$ opérations élémentaires.

Question 12 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.

Il suffit de remplacer, dans le calcul de ω_n (`wn` dans le code Python) le signe de l'argument de l'exponentielle.

```python
    wn = math.exp(+1j*2*math.pi/n) # + au lieu de - pour avoir la transformée inverse
```

2.B.2 Convolution avec la TFD

Question 13 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).

La question porte sur le produit terme à terme, et pas sur le produit de convolution qui est l'objet de la question suivante.

```python
def produit(x,h):
  # Retourne la liste contenant le produit terme à terme des listes x et h
  # on suppose que x et h sont de même longueur
  Y = []
  for k in range(len(x)):
    Y.append(x[k] * h[k])

  return Y
```

Question 14 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.

On attend une suite d'instructions, pas une fonction. D'après le sujet, le produit de convolution correspond à un produit terme à terme dans le domaine fréquentiel, il suffit donc de passer dans ce domaine.

```python
# On passe dans le domaine fréquentiel
X = Fourier_directe(x)
H = Fourier_directe(h)

# Produit de convolution = produit terme à terme dans le domaine fréquentiel
Y = produit(X,H)

# On revient dans le domaine temporel
y = Fourier_directe_inverse(Y)
```

Question 15 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.

Soit `n` la dimension de `x` et de `h`. Les appels à `Fourier_directe(x)`, `Fourier_directe(h)` ont une complexité en $O(n^2)$ opérations élémentaires, tout comme `Fourier_directe_inverse(Y)` (`Y` est de longueur n). L'appel à `produit(X,H)` a une complexité en $O(n)$ (X et H sont de longueur n).

La complexité asymptotique de la détermination d'un produit de convolution en utilisant la transformée de Fourier discrète est donc en $\boxed{O(n^2)}$ opérations élémentaires.

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

Question 16 Montrer en justifiant précisément votre réponse que l'algorithme conduit par les appels récursifs de la fonction `Fourier_rapide(x,N)` termine.

La fonction `Fourier_rapide(x,N)` ne fait aucun appel à elle-même si et seulement si son second argument $N = 1$. Ce second argument doit être un nombre entier, puisqu'il est utilisé comme indice dans une liste.

Si $N > 1$, alors elle réalise 2 appels à elle-même avec pour second argument $N_2 = N//2$ (avec // l'opérateur de quotient de la division euclidienne, ou division entière), c'est donc bien une fonction récursive.

Si $N = 1$: l'algorithme récursif termine (puisqu'aucun appel récursif n'est fait). Soit $N \in \mathbb{N}^* \setminus \{1\}$. On suppose que l'algorithme termine $\forall N' \in [\![1, N-1]\!]$. Comme $N > 1$, alors $N//2 < N$ et $N//2 \geq 1$ donc $N//2 \in [\![1, N-1]\!]$: l'algorithme récursif termine donc. Donc l'algorithme récursif termine $\forall N \in [\![1, N]\!]$.

Par récurrence, l'algorithme récursif termine $\forall N \in \mathbb{N}^*$

En revanche, l'algorithme récursif ne termine pas si $N = 0$, mais cette valeur n'a pas d'intérêt, puisque N correspond au nombre de coefficients de fréquences que l'on souhaite obtenir.

Question 17 Ecrire une fonction `nextpow2(n)` qui retourne N tel que défini ci-dessus.

Plusieurs implémentations sont possibles. En cherchant p :

```python
def nextpow2(n):
  p = 0
  while 2**p <= n:
    p += 1
  return 2**(p-1)
```

En cherchant directement N :

```python
def nextpow2(n):
  N=1
  while N<=n:
    N=2*N
  return N//2
```

Ou, si les fonctions `math.log2` et `math.ceil` sont autorisées :

```python
def nextpow2(n):
  return 2**(math.ceil(math.log2(n)))
```

Question 18 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 ?

Soit nbm(p) le nombre de multiplications réalisées lors d'un appel de la fonction `Fourier_rapide(x,`$N = 2^p$`)`. On a nbm(0) = 0.

$$\mathrm{nbm}(p) = \underbrace{\mathrm{nbm}(p-1)}_{\text{appel récursif termes impairs}} + \underbrace{\mathrm{nbm}(p-1)}_{\text{appel récursif termes pairs}} + \underbrace{2}_{\mathrm{w}\,:\,2\times j\times pi} + \underbrace{3\times 2^{p-1}}_{\text{boucle for}}$$

$$\mathrm{nbm}(p) - 2\cdot\mathrm{nbm}(p-1) = 3\cdot 2^{p-1} + 2$$

On obtient une équation récurrente linéaire d'ordre 1 non-homogène. En décalant d'un indice :

$$\mathrm{nbm}(p-1) - 2\cdot\mathrm{nbm}(p-2) = 3\cdot 2^{p-2} + 2$$

Puis en soustrayant la dernière équation à l'avant dernière :

$$\mathrm{nbm}(p) - 2\cdot\mathrm{nbm}(p-1) - \mathrm{nbm}(p-1) + 2\cdot\mathrm{nbm}(p-2) = 3\cdot 2^{p-1} - 3\cdot 2^{p-2}$$

$$\mathrm{nbm}(p) - 3\cdot\mathrm{nbm}(p-1) + 2\cdot\mathrm{nbm}(p-2) = \frac{3}{4}\cdot 2^p$$

On décale de nouveau d'un indice, et on soustrait 2 fois cette nouvelle équation à la précédente :

$$\mathrm{nbm}(p-1) - 3\cdot\mathrm{nbm}(p-2) + 2\cdot\mathrm{nbm}(p-3) = \frac{3}{4}\cdot 2^{p-1}$$

$$\mathrm{nbm}(p) - 5\cdot\mathrm{nbm}(p-1) + 8\cdot\mathrm{nbm}(p-2) - 4\cdot\mathrm{nbm}(p-3) = \frac{3}{4}\cdot 2^p - 2\cdot\frac{3}{4}\cdot 2^{p-1} = 0$$

Équation caractéristique : $r^3 - 5\cdot r^2 + 8\cdot r - 4 = 0 = (r-2)^2(r-1)$

La solution est donc de la forme $a\cdot 2^p + b\cdot 2^p\cdot p + c\cdot 1^p = a\cdot 2^p + b\cdot p\cdot 2^p + c$

On cherche à déterminer a, b et c, en substituant la forme de la solution dans l'équation récurrente :

$$a\cdot 2^p + b\cdot p\cdot 2^p + c - 2\cdot(a\cdot 2^{p-1} + b\cdot(p-1)\cdot 2^{p-1} + c) = 3\cdot 2^{p-1} + 2$$

$$2^{p-1}\cdot(2\cdot a + 2\cdot b\cdot p - 2\cdot a - 2\cdot b\cdot p + 2\cdot b) + c - 2\cdot c = 3\cdot 2^{p-1} + 2$$

$$2^{p-1}\cdot(2\cdot b) - c = 3\cdot 2^{p-1} + 2$$

Donc $c = -2$ et $b = \frac{3}{2}$ (par identification des coefficients). De plus, on a :

$$\underbrace{\mathrm{nbm}(0) = 0}_{\text{condition initiale}} = a\cdot 2^0 + \frac{3}{2}\cdot 0\cdot 2^0 - 2 = a - 2, \text{ donc } a = 2$$

$$\boxed{\mathrm{nbm}(p) = 2\cdot 2^p + \frac{3}{2}\cdot p\cdot 2^p - 2 = 2^{p-1}(3p+4) - 2}$$

On a donc $\boxed{\frac{3\times N\times log_2(N)}{2} + 2\times N - 2}$ multiplications réalisées. La complexité asymptotique en fonction de N est alors en $\boxed{O(N\times log(N))}$ opérations élémentaires.

Question 19 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.

	$N = 10^3$	$N = 10^4$	Augmentation par décade	Ordre de complexité
Convolution directe	0,7 s	70 s	×100	$O(N^2)$
Convolution TFD	8 s	800 s	×100	$O(N^2)$
Convolution FFT	0,035 s	0,45 s	×13	$O(N \times log(N))$

Le temps d'exécution par convolution directe est multiplié par 100 lorsque N est multiplié par 10 (on peut le remarquer en relevant les temps d'exécution pour $N = 10^3 \rightarrow 0{,}7\,\mathrm{s}$ et $N = 10^4 \rightarrow 70\,\mathrm{s}$). L'ordre de complexité est donc en $O(N^2)$ opérations élémentaires.

On remarque que la courbe qui concerne la simulation par convolution par transformée de Fourier discrète forme une droite parallèle à celle par convolution directe : l'ordre de complexité est donc identique, en $O(N^2)$ opérations élémentaires. Seul le coefficient devant le N^2 est différent (il est plus important, puisque les temps à tailles des données identiques sont plus grands), mais cela n'a pas de conséquence sur l'ordre de complexité.

Enfin, le temps d'exécution de la simulation par convolution par transformée de Fourier rapide est multiplié par 13 lorsque N est multiplié par 10 (de 10^3 à 10^4) : l'ordre de complexité semble donc être en $O(N \times log(N))$: la multiplication par 10 donne une complexité d'ordre N, tandis que la multiplication par 1,3 correspond à l'augmentation de $log(N)$ entre 10^3 et 10^4 : $\frac{log 10^4}{log 10^3} = \frac{4}{3} \approx 1,33$.

Ces trois ordres de complexité évalués graphiquement sont cohérents avec les résultats obtenus questions 11, 15 et 18. Ils permettent de conclure que l'utilisation de la transformée de Fourier est pertinente en temps de calcul pour réaliser un produit de convolution, mais seulement si l'on utilise la transformée de Fourier rapide : la transformée de Fourier discrète n'apporte aucun gain en complexité, et est même plus coûteuse en temps de calcul d'un facteur 10 d'après la figure 7.

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

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

Question 20 Comment justifier le choix de la fréquence $fech = 44\,100\,\mathrm{Hz}$ pour échantillonner un signal audible ?

La fréquence d'échantillonnage $fech = 44\,100\,\mathrm{Hz}$ permet d'enregistrer tous les sons du domaine audible, puisqu'il est possible d'enregistrer un signal sinusoïdal jusqu'à la fréquence (exclue) de $22\,050\,\mathrm{Hz}$ (d'après la condition de Nyquist-Shannon), ce qui permet donc d'avoir tout le domaine $[20\,\mathrm{Hz}; 20\,\mathrm{kHz}]$.

Question 21 Quel est l'intérêt principal du code complément à 2 pour représenter des entiers signés ?

Le code complément à 2 permet d'effectuer simplement les opérations arithmétiques (selon les mêmes méthodes que les nombres codés en binaire naturel).

Par exemple, sur 4 bits : $(1)_{10} + (-1)_{10} = (0001)_{c2} + (1111)_{c2} = (0000)_{c2} = (0)_{10}$

Question 22 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.

Le plus grand entier positif représentable sur 16 bits sans complément à 2 (donc en binaire naturel) est $2^{16} - 1 = 65535$.

En complément à 2 sur 16 bits, le plus grand entier positif représentable est $2^{15} - 1 = 32767$.

En complément à 2 sur 16 bits, le plus petit entier négatif est $-2^{n-1} = -32768$.

Question 23 Que retourne la commande suivante :

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

$$(-32768)_{10} = (80\ 00)_{16} big\ endian = (00\ 80)_{16} little\ endian$$

Donc la commande renvoie `b'\x00\x80'`.

Question 24 Que retourne la commande suivante :

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

$$(2A\ 00)_{16} little\ endian = (00\ 2A)_{16} big\ endian = (42)_10$$

Donc la commande renvoie $\boxed{42}$.

Question 25 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 ?

L'opérateur + utilisé sur deux variables de type `bytes` concatène les représentations hexadécimales, il ne réalise pas une addition des deux nombres. L'opération réalisée est donc une $\boxed{\text{concaténation}}$.

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

Question 26 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.

```python
# duree * frequence = len(voieG), donc i va bien parcourir le domaine des indices de la liste voieG
g = 32767 * voieG[i] / Maxi
# g sera bien compris dans [-32768;32767], car Maxi > MaxiG qui est un majorant des valeurs absolues des valeurs de voieG
# on multiplie par 32767 (et non 32768) pour éviter d'obtenir la valeur +32768
gbyte = int.to_bytes(g,2,byteorder='little',signed=True)
# idem pour la voie droite
d = 32767 * voieD[i] / Maxi
dbyte = int.to_bytes(d,2,byteorder='little',signed=True)
```

Question 27 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 question Q25 notamment.

```python
# on écrit la frame WAVE en concaténant la valeur gauche et la valeur droite
wavef.writeframesraw(gbyte+dbyte)
```

Question 28 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`.

```python
def addzeros(voie,N):
  # Pour ne pas modifier en place, la liste est copiée
  voie2 = voie[:]

  while len(voie2) < N:
    voie2.append(0)

  return voie2
```

Question 29 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`.

```python
def convolution_reverb(fichier1,fichier2,fichier3):

  # Récupération des deux signaux
  signal1 = extrawave_2oct(fichier1)
  signal2 = extrawave_2oct(fichier2)

  # Détermination de la longueur du signal à obtenir
  # Par principe des fichiers WAVE, les voies gauche et droite
  # d'un même fichier ont la même longueur, N sera donc le même
  N = nextpow2(len(signal1[0]) + len(signal2[0] - 1)

  # Allongement des signaux
  g1,d1 = addzeros(signal1[0],N), addzeros(signal1[1],N))
  g2,d2 = addzeros(signal2[0],N), addzeros(signal2[1],N))

  # Produit de convolution à gauche et à droite
  signal3 = []
  for voie in [(g1,g2),(d1,d2)]:
    F1 = Fourier_rapide(g1,N)
    F2 = Fourier_rapide(g2,N)
    F3 = produit(F1,F2)
    f3 = Fourier_rapide_inverse(F3,N)
    signal3.append(f3)

  # Ecriture du fichier résultat
  creawave_2oct(signal3[0],signal3[1],fichier3)

  # La fonction ne renvoie rien
```

Question 30 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.

Sur la figure 8, on observe des sons distinctement séparés par des moments de silence (3 sons sur la voie gauche, 5 sur la voie droite). Sur la figure 13, on observe ces mêmes sons (avec des amplitudes plus élevées que le reste du signal), mais ils ne sont plus séparés distinctement par du silence : la réverbération des sons crée du bruit sur les temps de silence qui suivent.

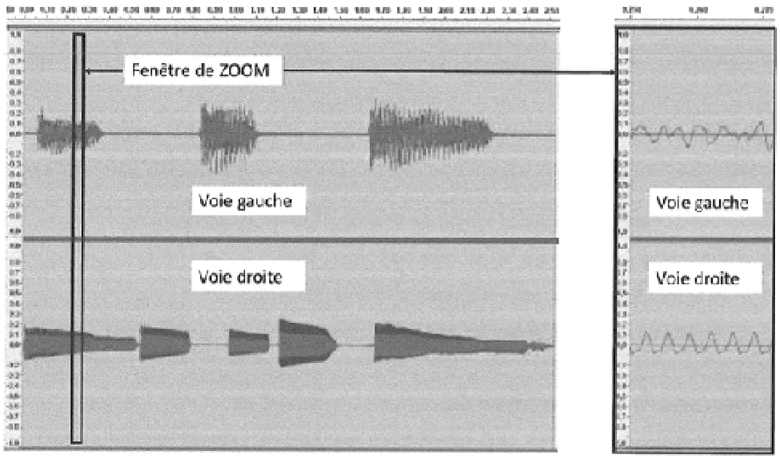


Figure 8 : Signal audio numérique stéréo

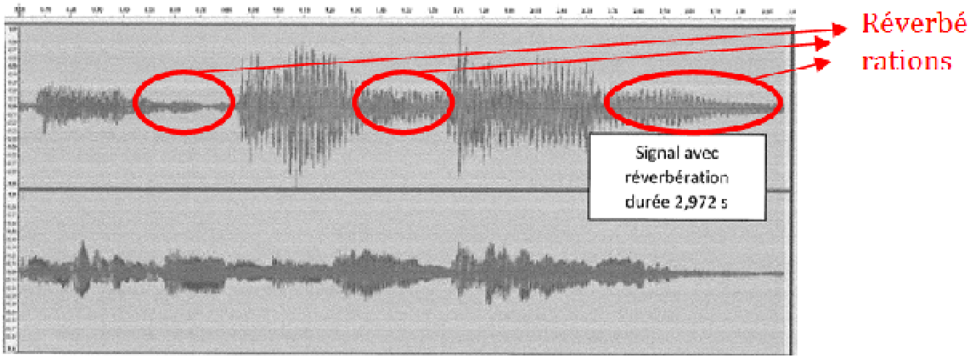


Figure 13 : signal avec réverbération