TP 4C – Electronique numérique

Problématiques :

- En quoi la numérisation affecte-t-elle un signal ? Quelles conditions sont à respecter pour minimiser la perte d'informations lors de la numérisation ?
- Avec un programme Python, comment exploiter la Transformée de Fourier Discrète pour accéder au spectre d'un signal s(t) après l'avoir échantillonné ?

Compétences expérimentales au programme :

Analyse spectrale. Échantillonnage, fréquence d'échantillonnage. Conséquences expérimentales du théorème de Nyquist-Shannon.	Réaliser l'échantillonnage d'un signal. Commenter la structure du spectre du signal obtenu après échantillonnage. Mettre en évidence le phénomène de repliement du spectre provoqué par l'échantillonnage avec un oscilloscope numérique ou une carte d'acquisition. Choisir les paramètres d'une acquisition numérique destinée à une analyse spectrale afin de respecter la condition de Nyquist-Shannon, tout en optimisant la résolution spectrale. Capacité numérique : calculer, à l'aide d'un langage de programmation, la transformée de Fourier discrète d'un signal numérique.
Électronique numérique. Filtrage numérique.	Mettre en œuvre un convertisseur analogique/numérique et un traitement numérique afin de réaliser un filtre passe- bas ; utiliser un convertisseur numérique/analogique pour restituer un signal analogique. Capacité numérique : à l'aide d'un langage de programmation, simuler un filtrage numérique et visualiser son action sur un signal périodique.

Objectifs :

1. Mettre en évidence l'influence de la fréquence d'échantillonnage et énoncer la condition de Nyquist-Shannon.
2. Comprendre le principe de la quantification (pas de conversion / résolution en tension).
3. Calculer la TFD d'un signal à valeurs réelles (généré sous Python ou issu d'une acquisition) en utilisant la fonction rfft de la bibliothèque numpy.fft
4. Mettre en œuvre un filtre numérique sous Latis Pro ou avec un programme Python (vérifier l'effet du filtrage en comparant les spectres des signaux d'entrée et de sortie obtenus avec la fonction rfft).

Rotation TP4 :

	01-oct	08-oct	15-oct
Lazare - Antoine D.	C	B	A
Antoine V. - Stanislas	C	B	A
Zaccharie - Camille	B	A	C
Robin - Ilyès	A	C	B
Gabriel	A	C	B

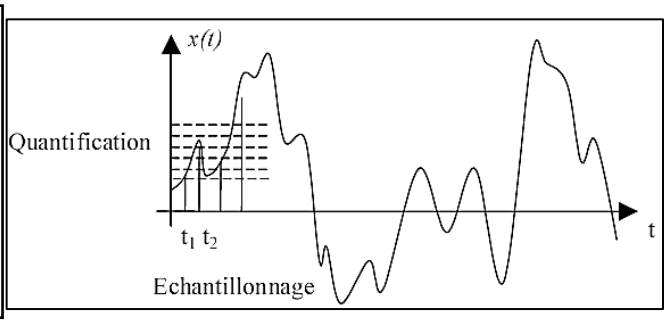
A faire pour la 1e séance de TP :
Lire § A à C et répondre aux questions ✎.
A faire pour le 05/11 :
Fin TP4B (résolution numérique équation diffusion thermique) + Lire § D et répondre aux questions ✎.

INTRO – Principe de la numérisation d'un signal

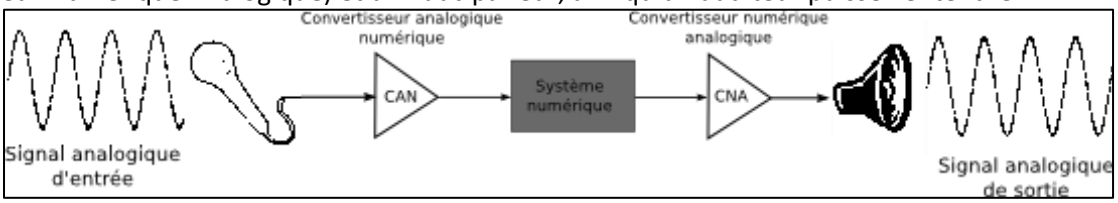
Les grandeurs physiques sont de natures diverses : P, T, pH… Lorsqu'on veut les traiter, on commence par **les convertir en un signal électrique via un capteur**. Comme la grandeur physique de départ dépend généralement du temps, le signal électrique obtenu est une fonction continue du temps : $x(t)$. Le signal $x(t)$ est qualifié d'**analogique** par opposition aux signaux **logiques/numériques**.

Lorsqu'on souhaite transmettre de l'information (voix, image…), ou la stocker ou la travailler, le format analogique devient vite limitant et/ou peut poser des problèmes de reproductibilité. Il est donc souvent nécessaire de **numériser le signal analogique** i.e. de le **convertir en un signal numérique**. Numériser $x(t)$ sur une durée finie permet de passer d'un **ensemble non dénombrable de valeurs réelles** à **un ensemble fini de valeurs numériques**, représentées par une succession de « 0 » et de « 1 ».

La conversion analogique → numérique d'un signal correspond à la succession de 2 étapes :
- **l'échantillonnage** qui permet de prélever un ensemble de valeurs prises à des instants discrets :
$\{t_k = kT_e = k/f_e\}$ avec f_e la fréquence d'échantillonnage,
- la **quantification** qui alloue à chacun de ces échantillons une valeur approchée car codée sur un nombre entier n de bits.



Ex : Dans la chaîne ci-dessous, on dispose initialement d'un son analogique que l'on enregistre via un microphone et un **CAN** (Convertisseur Analogique Numérique). Ce signal numérique peut subir divers traitements à travers un système numérique. Enfin, le signal numérique traité est restitué en signal sonore analogique, via un **CNA** (Convertisseur Numérique Analogique) et un haut-parleur, afin qu'un auditeur puisse l'entendre.



A) Echantillonnage

1) Principe

Le principe de l'échantillonnage est le suivant :
Le CAN génère une tension $w(t)$ (cf fig.1) avec T_e la période d'échantillonnage et τ une durée très faible devant T_e. Soit k un entier, $w(t)$ est tel que si $t \in [kT_e, kT_e + \tau]$ alors $w(t) = 1$, sinon $w(t) = 0$. Pour $\tau \to 0$, $w(t) = 1$ ssi $t = kT_e = t_k$ (cf intro p.1) : on parle de peigne de Dirac.
On veut convertir le signal analogique $x(t)$ représenté en pointillés sur la fig.2. Le signal enregistré par le CAN est $x_e(t) = x(t).w(t)$. **Pour $\tau \to 0$,** $x_e(t) = \{x(0), x(T_e), x(2T_e), \dots\}$ i.e. que le signal $x_e(t)$ est un ensemble dénombrable de valeurs réelles.

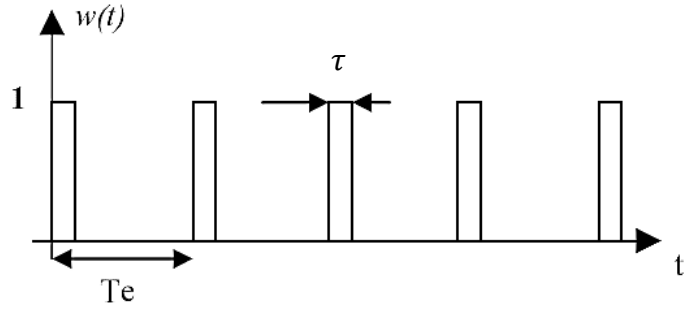


Fig.1

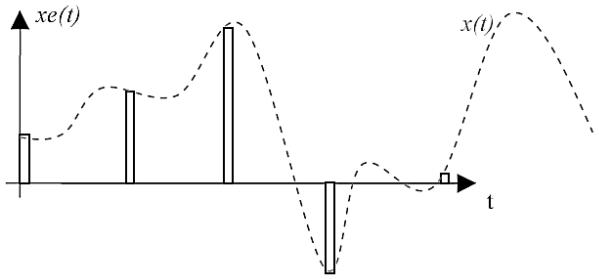
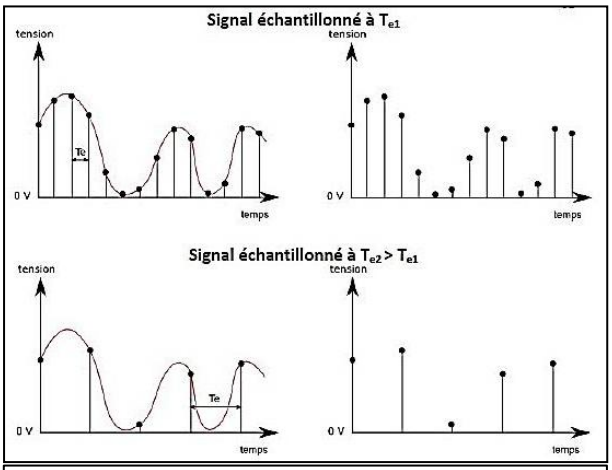


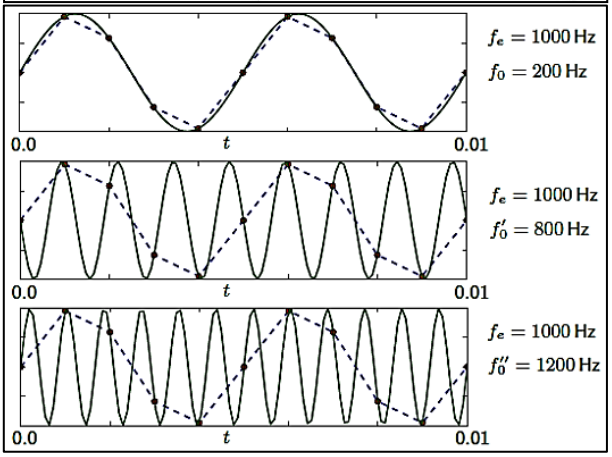
Fig.2

♦ Exemples d'échantillonnage d'un signal avec deux fréquences d'échantillonnage différentes : $f_{e2} < f_{e1}$.

Dans le 1er exemple, la fréquence d'échantillonnage choisie permet de reproduire les variations du signal. Ce qui n'est pas le cas du 2e exemple : il est clair que les échantillons recueillis ne sont pas suffisants pour reconstruire le signal d'origine.



♦ Exemples de 3 signaux analogiques différents donnant lieu au même signal échantillonné :
On représente en trait plein les signaux analogiques et en pointillés les signaux échantillonnés.



CCL du § A :
f_e doit être maximale pour que le signal échantillonné soit fidèle au signal de départ.
En pratique, les dispositifs d'acquisition sont **limités en fréquence** et **un nombre d'échantillons trop élevé peut poser problème** : temps de calcul, volume de stockage de données...
Dans la suite, on introduit une valeur minimale de f_e en dessous de laquelle, la numérisation pose problème.

2) Choix de la fréquence d'échantillonnage – Analyse spectrale d'un signal échantillonné

a) Paramètres temporels d'acquisition

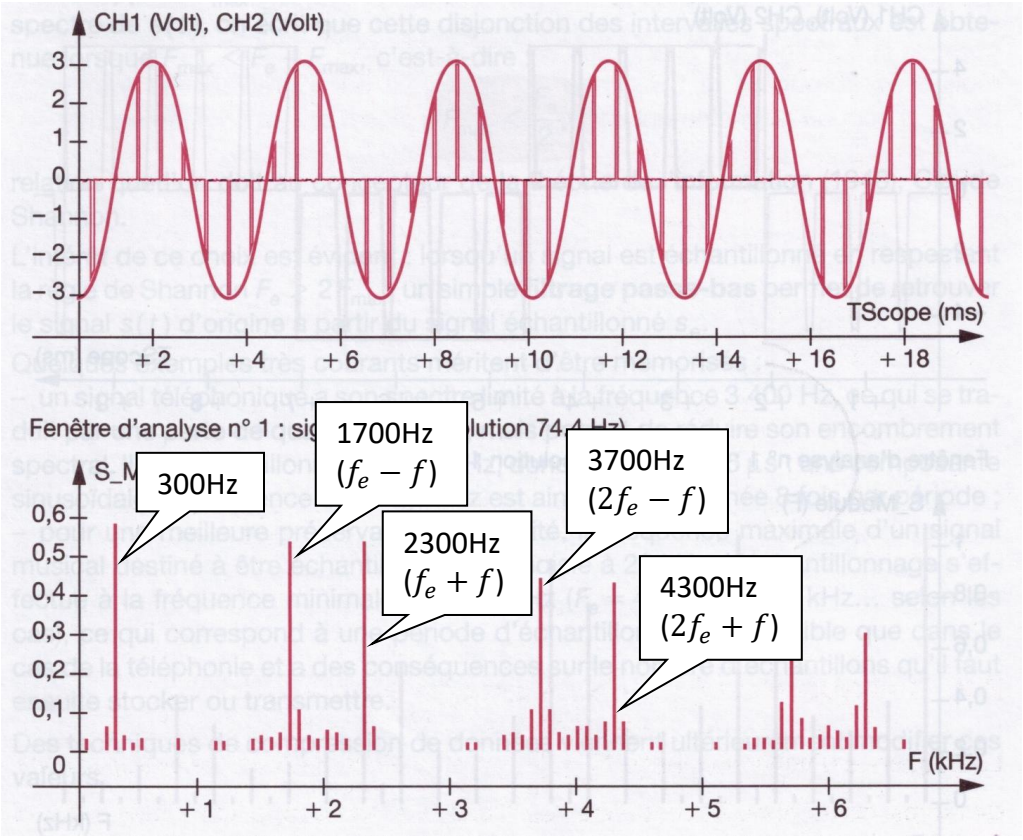
Pour réaliser une acquisition, il faut choisir :
- la durée totale d'acquisition, notée Total
- la période d'échantillonnage, notée T_e
- le nombre de points, noté Points

✎➲ 1. Rappeler la relation entre ces 3 grandeurs.

b) Fréquences répliquées – Condition de Nyquist-Shannon

♦ Les oscillogrammes ci-dessous correspondent à un signal sinusoïdal $x(t)$ de fréquence $f = 300\,Hz$ échantillonné à la fréquence $f_e = 2\,kHz$.

Dans le spectre du signal échantillonné $x_e(t)$, on retrouve la fréquence fondamentale $f = 300\,Hz$ du signal $x(t)$. Cependant, l'opération d'échantillonnage fait apparaitre d'autres fréquences : à 1700 Hz et à 2300 Hz, soit $f_e - f$ et $f_e + f$; et autour des multiples de la fréquence d'échantillonnage f_e : $nf_e - f$ et $nf_e + f$ avec n entier.



→ Interprétation mathématique des fréquences répliquées :

Le signal (t) , de période T_e et de fréquence $f_e = \frac{1}{T_e}$, est décomposable en série de Fourier :

$$w(t) = \sum_{n=0}^{\infty} a_n \cos(2\pi n f_e t)$$

Soit un signal analogique $x(t)$ sinusoïdal :

$$x(t) = A\cos(2\pi f t)$$

On a

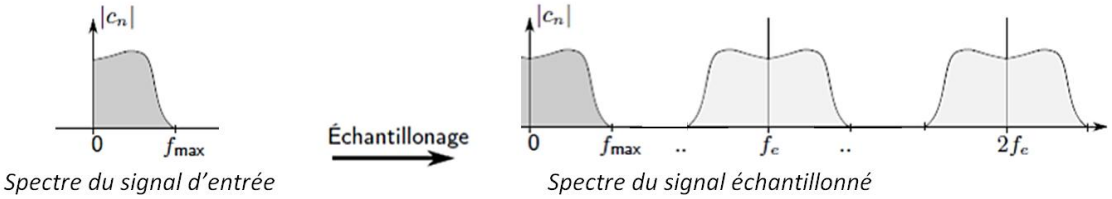
$$x_e(t) = x(t).w(t) = \sum_{n=0}^{\infty} \left(a_n \cos(2\pi n f_e t)\,.\,A\cos(2\pi f t)\right)$$

Avec

$$a_n \cos(2\pi n f_e t)\,.\,A\cos(2\pi f t) = a_n.A.\frac{1}{2}.\left(\cos(2\pi(nf_e + f)t) + cos(2\pi(nf_e - f)t)\right)$$

Ainsi le fait de multiplier le signal sinusoïdal $x(t)$ par l'harmonique de rang n du signal $w(t)$ engendre un **décalage et un dédoublement de sa fréquence f autour de nf_e** : il s'agit des **fréquences répliquées**.

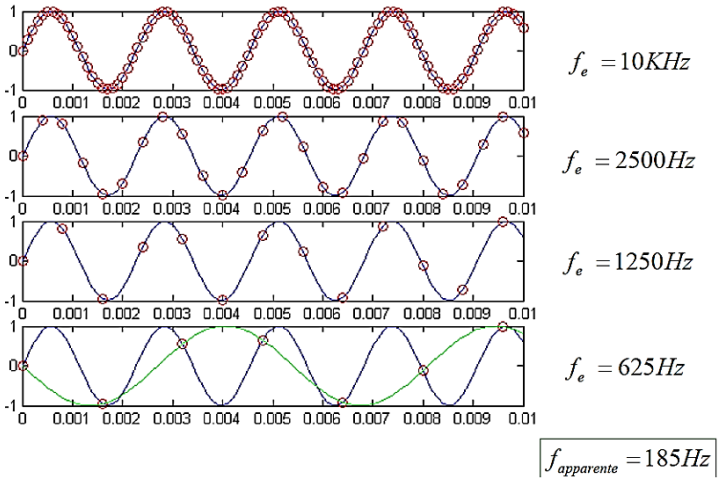
Ce résultat se généralise à tout signal analogique $x(t)$ ainsi l'allure du spectre du signal échantillonné $x_e(t)$ sera de la forme suivante :



Spectre du signal d'entrée — *Spectre du signal échantillonné*

→ Interprétation graphique de la fréquence répliquée $(f_e - f)$: fréquence apparente d'un signal échantillonné

On échantillonne un signal sinusoïdal correspondant au La-440, à différentes fréquences d'échantillonnage :



♦ Ces fréquences répliquées ne viennent pas empiéter sur le spectre originel du signal d'entrée ssi :

$f_e - f_{max} > f_{max}$ où f_{max} est la fréquence maximale du spectre du signal

soit $\boldsymbol{f_e > 2f_{max}}$.

A RETENIR : Condition de Nyquist-Shannon :

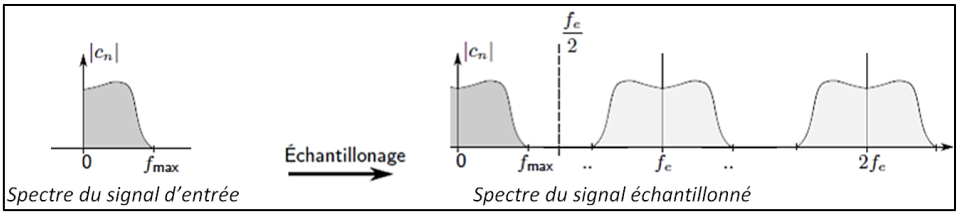
Au spectre du signal d'entrée, viennent s'ajouter des raies spectrales obtenues pas réplication des raies précédentes autour de valeurs multiples de la fréquence d'échantillonnage.

Ces fréquences répliquées n'empiètent pas sur le spectre originel du signal d'entrée ssi la **CONDITION DE NYQUIST-SHANNON** est respectée :

$$\boldsymbol{f_e > 2f_{max}}$$

Si $f_e > 2f_{max}$, un filtrage passe–bas de fréquence de coupure f_c telle que $f_{max} < f_c < \frac{f_e}{2}$ permet de retrouver le signal d'origine à partir du signal échantillonné.

Exemple d'échantillonnage qui respecte la condition de Nyquist-Shannon :



Spectre du signal d'entrée — *Spectre du signal échantillonné*

Intervalle de fréquences du spectre sous Latis Pro :

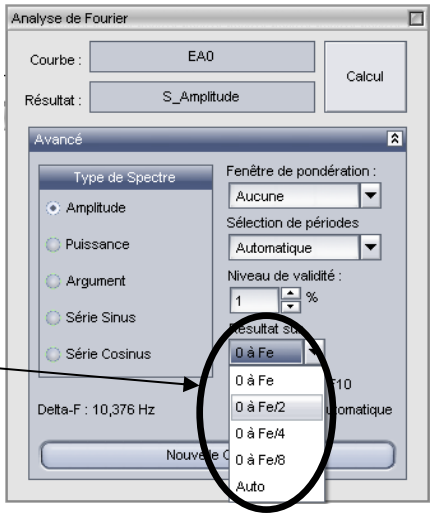
✋ Réaliser l'acquisition sur Latis Pro d'un signal sinusoïdal de fréquence $f = 400\ Hz$ en choisissant une fréquence d'échantillonnage $f_e = 10\ kHz$ et 10^3 points.

✎➲ 2. La condition Nyquist-Shannon est-elle respectée ?

✋ Réaliser le spectre du signal via l'onglet « traitements » puis « calculs spécifiques » et « Analyse de Fourier ».

Le logiciel propose différents intervalles d'affichage en fréquence.

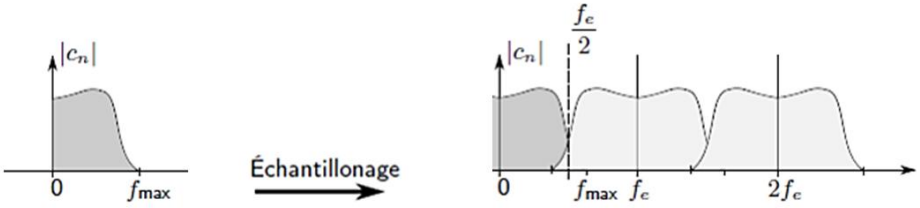
➲ 3. Pourquoi l'intervalle par défaut est-il : 0 à Fe/2 ?



c) Repliement du spectre

Si la condition de Nyquist-Shannon n'est pas respectée, les fréquences répliquées viennent empiéter sur le spectre originel du signal d'entrée : on parle de <u>REPLIEMENT SPECTRAL</u> (aliasing en anglais).

Exemple d'échantillonnage qui ne respecte pas la condition de Nyquist-Shannon :



Spectre du signal d'entrée *Spectre du signal échantillonné*

<u>Mise en évidence du repliement spectral</u> :

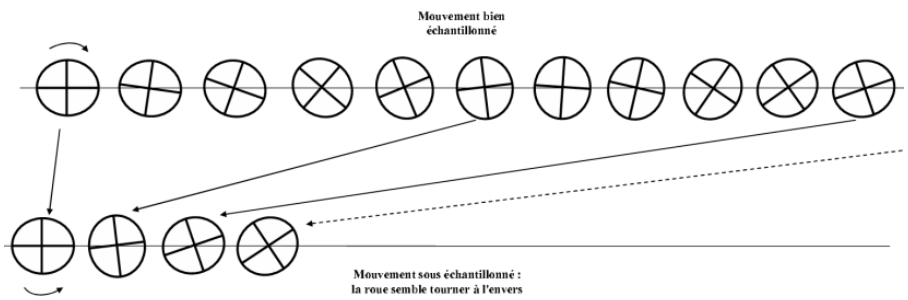
➲ 4. Réaliser l'acquisition sur Latis Pro d'un signal sinusoïdal de fréquence $f = 1kHz$ délivrée par le GBF en choisissant une fréquence d'échantillonnage de $f_e = \frac{4}{3}f$ (durée de l'acquisition : 100 ms). Commenter.

<u>Solution : Filtre anti-repliement</u>

Si on est limité en fréquence d'échantillonnage et que la condition de Nyquist-Shannon ne peut pas être vérifiée alors il faut filtrer le signal **en amont**, i.e. avant de le numériser, avec un filtre passe bas tel que $f_c < \frac{f_e}{2}$. Evidemment, la perte des hautes fréquences du spectre peut entraîner une perte d'informations.

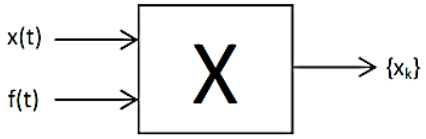
<u>Rq</u> : On peut mettre en évidence le phénomène de repliement spectral en regardant le film d'une roue de charrette qui tourne. On voit souvent apparaître une vitesse de rotation différente de la vitesse d'avancement réelle de la charrette. Parfois même, la roue semble aller en sens inverse.

Cela est dû au fait qu'un film est un échantillonnage d'un phénomène continu : 25 photos prises par seconde. Ainsi, il y a repliement spectral pour les phénomènes périodiques de fréquence supérieure à 12 Hz.

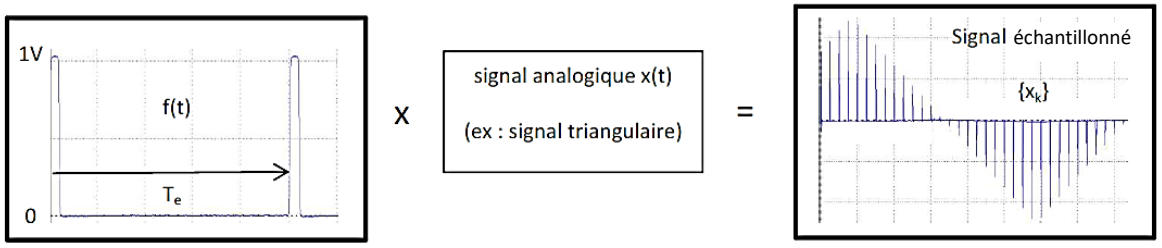


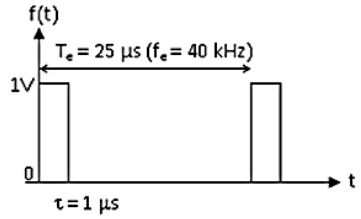
3) Réalisation d'un échantillonnage « analogique » - Mise en évidence des fréquences répliquées

Le principe de l'échantillonnage analogique consiste à multiplier le signal $x(t)$ par un train d'impulsions $f(t)$ de fréquence $f_e = \frac{1}{T_e}$ qui correspond à la fréquence d'échantillonnage, cf § A.1.



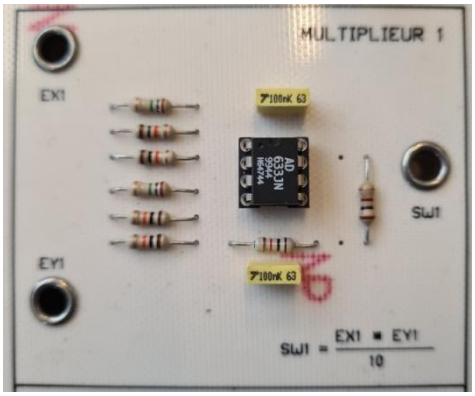
Ex :





✋ Régler un des GBF pour qu'il délivre le train d'impulsions : fonction « pulse », fréquence $f_e = 40$ kHz, durée de l'impulsion $\tau = 1\ \mu s$, tension minimale = 0 V et tension maximale = 1 V.

✋ Régler le 2e GBF pour qu'il délivre le signal $x(t)$ à échantillonner : signal triangulaire de fréquence $f = 1$ kHz, de valeur moyenne nulle, d'amplitude maximale (accessible avec le GBF).



✋ Utilisation du multiplieur :

- **Alimenter la plaquette en ± 15 V, relier la borne 0 V de l'alimentation stabilisée à la masse du GBF.**
- Connecter le train d'impulsions $f(t)$ sur la voie EX1 et signal $x(t)$ à échantillonner sur la voie EY1.
- Le signal échantillonné $\{x_k\}$ correspond à la borne de sortie SW1.

✋ Visualiser sur l'oscilloscope :

- le signal $x(t)$ à échantillonner sur la voie 1 ;
- le train d'impulsions $f(t)$ sur la voie 2 ;
- le signal échantillonné $\{x_k\}$ sur la voie 3.

✋ Acquérir le signal échantillonné sur Latis Pro : N = 4000, T_e = 500 ns et Total = 2 ms.

✋ Faire le spectre du signal échantillonné.

➲ 5. Analyser les fréquences présentes dans le spectre (cf § A.2.b). Y a-t-il repliement spectral dans ce cas ?

B) Quantification

On appelle **pas de quantification** ou **pas de conversion** ou **résolution en tension** d'un CNA l'écart (en volt) entre deux valeurs binaires successives. On le note r dans la suite.

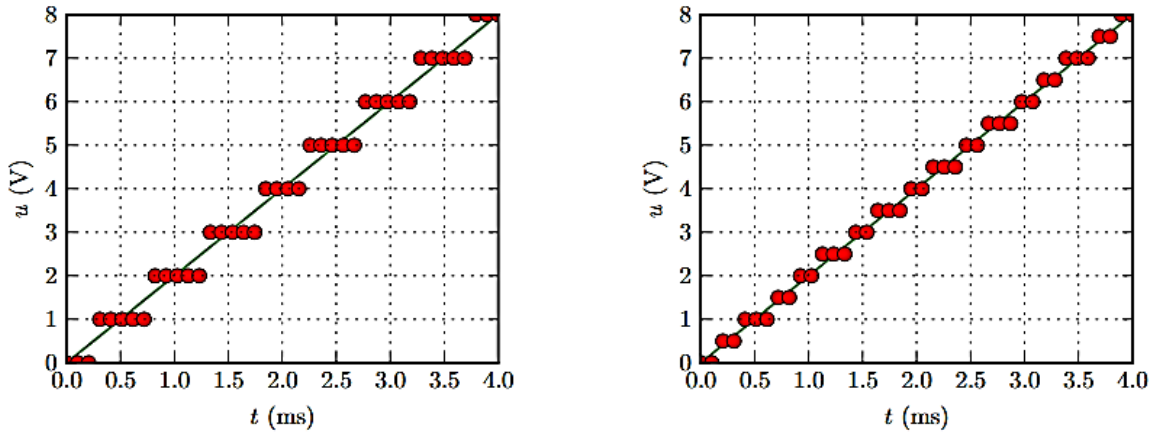


Figure 8 – Deux exemples de numérisation d'une même tension. Le signal analogique est représenté en trait plein bleu, le signal numérisé par les points . Dans les deux cas la période d'échantillonnage est de 0,1 ms. Sur la figure de gauche le pas de quantification vaut 1 V alors qu'il vaut 0,5 V sur la figure de droite.

1) Principe des conversions NA et AN – Réseau $R-2R$

On dispose d'une platine de conversion AN-NA constituée d'un réseau $R-2R$:

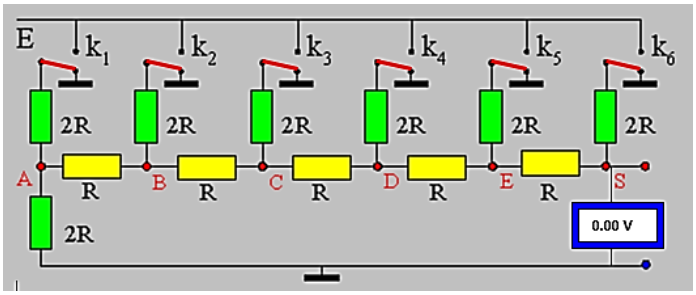


Schéma d'un réseau $R-2R$ à 6 bits
http://ressources.univ-lemans.fr/AccesLibre/UM/Pedago/physique/02/electro/cnar2r.html

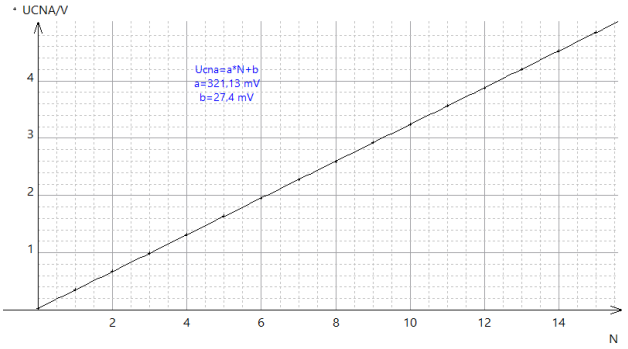
E est une tension de référence, pour la platine étudiée, $E = 5{,}12\ V$.

a) Conversion numérique analogique

Selon l'état des interrupteurs k_i (fermé sur la ligne de masse ou fermé sur la ligne de potentiel E), on obtient une tension différente en sortie.
L'état fermé sur la ligne de masse correspond au bit 0 et l'état fermé sur la ligne de potentiel E correspond au bit 1.
L'interrupteur noté k_1 sur la figure ci-dessus correspond au bit de poids le plus faible.

Avec un CNA 4 bits, on peut générer 2^4 = 16 valeurs différentes de tension en sortie entre 0 V et $U_{CNA\,max}$, cf graphe ci-contre.
Comme en témoigne la modélisation de la courbe, le CNA étudié est linéaire. La pente de la droite correspond à r la « résolution en tension » du CNA.
On peut vérifier que :

$$r = \frac{E}{2^4} \quad et\ U_{CNA\,max} = E - r$$



<u>Rq</u> : ces relations sont valables pour un réseau $R-2R$ de n bits avec n entier naturel quelconque.

<u>b) Conversion analogique numérique</u>

Il s'agit de comparer la tension analogique U à convertir à la tension de sortie U_{CNA} du réseau $R-2R$.
On dispose ici de deux modes de conversion AN.

i) Mode simple rampe

U_{CNA} est générée linéairement bit par bit.
La conversion est terminée dès que $U_{CNA} > U$.

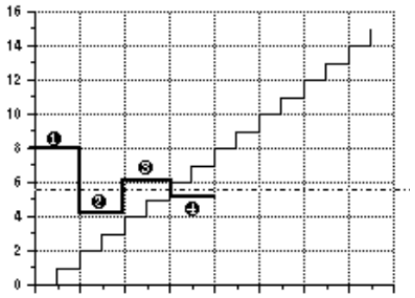
ii) Mode approximations successives

U_{CNA} est générée par dichotomie :

U est d'abord comparée à $\frac{U_{CNA\,max}}{2}$:

Si $U < \frac{U_{CNA\,max}}{2}$ alors U est comparée à $\frac{U_{CNA\,max}}{2} - \frac{U_{CNA\,max}}{4}$ …

Si $U > \frac{U_{CNA\,max}}{2}$ alors U est comparée à $\frac{U_{CNA\,max}}{2} + \frac{U_{CNA\,max}}{4}$ …



en pointillés tension U analogique à convertir
en trait fin tension U_{CNA} générée bit par bit (mode simple rampe)
en trait gras tension U_{CNA} générée par dichotomie (mode approximations successives)

Pour un convertisseur n bits, la conversion par dichotomie nécessite n étapes quelle que soit la valeur de la tension U analogique à convertir.
La conversion par simple rampe a une durée qui dépend de la valeur de la tension U analogique à convertir et elle est moins efficace que la conversion par dichotomie.

2) Etude de la conversion AN réalisée par la carte d'acquisition

→→→ A faire en dernier s'il vous reste du temps.

On se propose de déterminer le nombre de bits de la conversion AN réalisée par la carte d'acquisition EuroSmart en étudiant le signal numérique associé à un signal analogique continu.

L'échantillonnage est suivi par l'opération de quantification. On considère une conversion AN telle que les tensions sont codées sur n bits.
On note C le calibre choisi pour acquérir les tensions : la tension maximale que l'on peut acquérir vaut $+C$ et la tension minimale que l'on peut acquérir vaut $-C$. On note r la « résolution en tension » du CAN ou « pas de conversion », il s'agit de l'écart minimal entre deux valeurs possibles de tension.

➲ 6. Exprimer r en fonction de C et n.

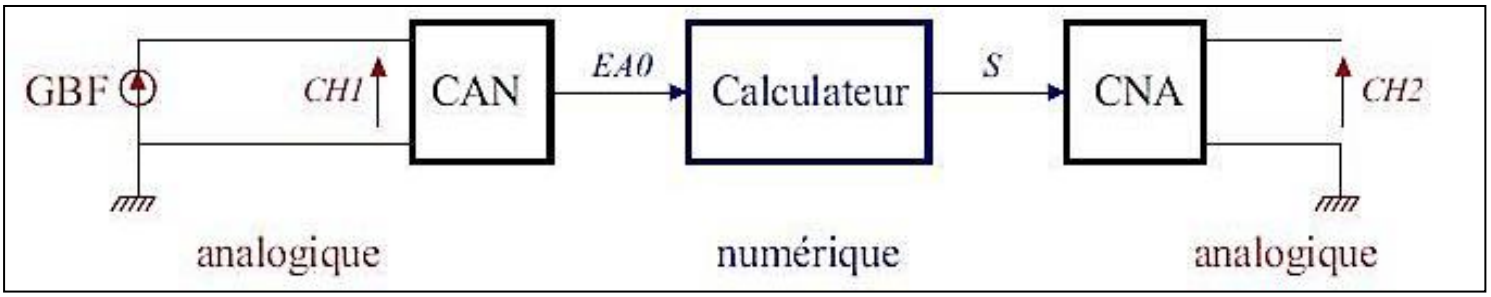
➲ 7. Réaliser l'acquisition d'un signal continu délivré par le GBF sur Latis Pro (période d'échantillonnage de 4 µs et durée d'acquisition de 1 s). Zoomer fortement sur une partie des points acquis pour visualiser et mesurer le pas de conversion r.

➲ 8. Sachant que le calibre de la carte EuroSmart est imposé à $\pm$ 10 V, déterminer le nombre de bits n de la conversion AN réalisée par la carte d'acquisition. La notice de la carte d'acquisition précise qu'elle est constituée d'un CAN 12 bits. Votre résultat expérimental est-il compatible avec cette information ?

C) Filtrage numérique avec la carte d'acquisition

L'un des principaux intérêts de la numérisation d'un signal est la simplicité d'un traitement numérique comparé à un traitement analogique : les tâches peuvent être plus complexes et plus facilement modifiables : changer la valeur d'une variable dans un programme est bien plus simple que de changer de composant électronique !
Une fois traité, le signal numérique peut être de nouveau converti en signal analogique par un CNA.

On se propose de numériser un signal, le filtrer numériquement (passe-bas) puis de le reconvertir sous forme analogique avec la carte Eurosmart.



CAN (convertisseur analogique **numérique**) : Entrée **EA0**
CNA (convertisseur **numérique** analogique) : Sortie **S**

a) Algorithme de filtrage

La fonction de transfert d'un filtre passe-bas s'écrit sous forme canonique :

$$\underline{H} = \frac{H_0}{1 + j\omega\tau}$$

Dans la suite, on prendra $H_0 = 1$ et $f_c = \frac{1}{2\pi\tau} = 100\ Hz$.

✎➲ 9. Donner l'équation différentielle entre les signaux de sortie $s(t)$ et d'entrée $e(t)$, nommé EA0 avec la carte.

Pour transformer cette relation différentielle en une équation sur les échantillons $s_n = s(t_n = nT_e)$ avec T_e la période d'échantillonnage, il faut approximer la dérivée $\frac{ds}{dt}$.
La correspondance qui permet de passer d'un filtre analogique à un filtre numérique n'est pas unique. Le plus simple est d'utiliser le schéma d'Euler explicite, le pas de temps étant égal à la période d'échantillonnage.
✎➲ 10. Donner l'expression approchée de la dérivée en fonction de la période d'échantillonnage T_e et des échantillons s_n et s_{n+1}.
✎➲ 11. En déduire une relation de récurrence permettant de calculer s_{n+1} à partir de $e_n = e(t_n = nT_e)$ et $s_n = s(t_n = nT_e)$.
Ainsi, on peut construire point par point le signal numérique s à partir de e.

b) Mise en œuvre – implémentation de l'algorithme sous Latis Pro

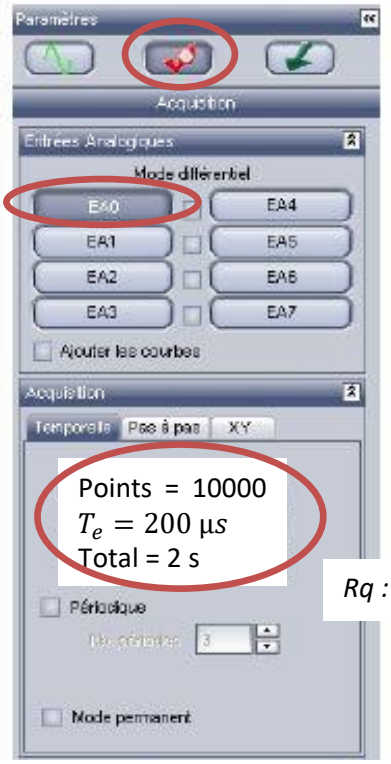
✋ Ouvrir une feuille de calcul via l'onglet « Traitement » et écrire le programme suivant :

```
Te=2e-4
fe=1/Te
fc=100
tau=1/(2*3.14*fc)
S=Table(0)
S=S[n-1]+Te/tau*(EA0[n-1]-S[n-1])
```

✋ Régler le GBF pour qu'il délivre un signal sinusoïdal de valeur moyenne nulle, d'amplitude 2 V et de fréquence $f = 10$ Hz dans un 1er temps. Visualiser ce signal à l'oscilloscope (CH1) et acquérir ce signal sur l'entrée EA0 de la carte d'acquisition qui le convertit en signal numérique.

On règle l'acquisition comme ci-contre.

La sortie numérique S peut être visualisée sur Latis Pro. La sortie **analogique** correspondante est visualisée en connectant la voie CH2 de l'oscilloscope à la sortie SA1 de la carte d'acquisition.





$Rq : T_e \ll \tau \ll Total$

➲ 12. Faire l'acquisition pour une fréquence du GBF f = 10 Hz. Observer sur Latis Pro les signaux numériques EA0 et S. Vérifier que, comme pour un filtre analogique, le régime permanent s'établit au bout d'un certain temps (quelques τ).

➲ 13. Lancer l'acquisition pour un signal d'entrée de fréquence f = 10 Hz puis 100 Hz puis 500 Hz et 4930 Hz. Pour chaque fréquence, commenter les signaux obtenus.

NB : On reviendra sur le filtrage numérique au TD/DM.

D) TFD

1) De la Transformée de Fourier à la Transformée de Fourier Discrète

♦ La **Transformée de Fourier** (TF) $S(f)$ de la fonction $s(t)$ s'obtient en calculant l'intégrale :

$$S(f) = \int_{-\infty}^{+\infty} s(t) \cdot \exp(-i2\pi ft) \cdot dt \qquad \text{avec } i^2 = -1$$

La composante $S(f)$ est un nombre complexe, on représente en général $|S(f)|$.
Les fréquences $f \in \mathbb{R}$ dans le cas général mais les valeurs $f \geq 0$ suffisent pour décrire un signal réel.

♦ Lorsqu'on étudie un signal $s(t)$ en TP, on réalise l'acquisition de ce signal sur une **durée T_{acq} finie**. On fait alors l'approximation suivante pour la TF :

$$S(f) \approx \int_{0}^{T_{acq}} s(t) \cdot \exp(-i2\pi ft) \cdot dt$$

Cette approximation a pour conséquence un **élargissement des raies** : dans le cas d'un signal purement sinusoïdal de fréquence f_1, le spectre comporte une raie centrée sur f_1 et de largeur $\Delta f \approx \frac{1}{T_{acq}}$.

Rq : on exploitera ce résultat lorsqu'on étudiera le paquet d'ondes en EM et les trains d'ondes en optique.

♦ Lorsqu'on fait l'acquisition du signal $s(t)$ en **l'échantillonnant avec une période d'échantillonnage T_e** alors on obtient un signal discret dont les termes sont :

$$s_n = s(t_n) \text{ avec } t_n = n \cdot T_e \text{ pour } n \in [\![0, N-1]\!] \text{ avec } N = \frac{T_{acq}}{T_e}$$

Dans ce cas, on approxime la TF par la **somme discrète** suivante (méthode des rectangles à gauche) :

$$S(f) \approx T_e \cdot \sum_{n=0}^{N-1} s_n \cdot \exp(-i2\pi f t_n) = T_e \cdot \sum_{n=0}^{N-1} s_n \cdot \exp(-i2\pi f n T_e)$$

En considérant la liste de fréquences suivante :

$$f_k = k \cdot \frac{1}{T_{acq}} = \frac{k}{NT_e} = \frac{k}{N} f_e \text{ pour } k \in [\![0, N-1]\!]$$

avec f_e la fréquence d'échantillonnage

on introduit la **Transformée de Fourier Discrète** (TFD) du signal échantillonné qui correspond à une liste dont les termes complexes sont définis par :

$$\underline{c_k} = \sum_{n=0}^{N-1} s_n \cdot \exp\left(-i2\pi \frac{n \cdot k}{N}\right) \text{ pour } k \in [\![0, N-1]\!]$$

Rq : $\underline{c_0} \in \mathbb{R}$ *et* $\underline{c_{N-k}} = \underline{c_k}^*$

Conformément au programme, on utilisera la fonction **rfft** de la bibliothèque **numpy.fft** sous python pour calculer la TFD d'un signal à valeurs réelles (r pour réel et fft pour **Fast Fourier Transform**[*]). On peut obtenir la liste des fréquences à l'aide de **rfft.rfftfreq** ou générer une liste des $f_k = \frac{k}{NT_e} = \frac{k}{N} f_e$ (cf §2).

[*] Le calcul de la TFD est fortement accéléré quand on utilise un algorithme appelé FFT pour Fast Fourier Transform qui a été mis au point dans les années 1960 par J.W. Cooley et John Tukey. L'algorithme FFT

réduit le nombre d'opérations pour passer d'une complexité temporelle en $O(N^2)$ (calcul de N sommes de N termes) à une complexité en $O(Nln(N))$, cf annexe.
C'est aussi l'algorithme FFT qui est utilisé par les oscilloscopes numériques et les logiciels de traitements de données, tels que LatisPro, pour obtenir le spectre d'un signal.
NB : Pour optimiser un très grand nombre de traitements, on choisit un **nombre de points d'acquisition** $\boldsymbol{N = 2^p}$ (une puissance de 2).

Pour N pair, l'algorithme FFT de calcul de la TFD donne l'évaluation suivante du spectre de $s(t)$:

- la composante continue du signal $s(t)$ est : $\frac{1}{N}\underline{c_0}$
- pour $k \in [\![0, \frac{N}{2}]\!]$, l'amplitude de la composante de fréquence $f_k = k \cdot \frac{1}{T_{acq}} = \frac{k}{N} f_e$ du signal $s(t)$ est : $\frac{2}{N}\left|\underline{c_k}\right|$

Ce spectre calculé se rapproche du spectre réel du signal $s(t)$ dans la mesure où les conditions suivantes sont respectées : **critère de Nyquist-Shannon, échantillonnage d'un signal périodique sur un nombre entier de périodes**.

2) Mise en œuvre : Calcul de la TFD d'un signal

On souhaite obtenir le spectre d'un signal via un programme Python utilisant les **fonctions rfft et rfftfreq de la bibliothèque numpy.fft**. Les documentations de ces fonctions sont fournies dans les doc 1 et 2 p. 18.
Répondre aux questions ci-après en complétant le programme Python « TP4C_rfft_à compléter » disponible sur Cahier de Prépa.

➲ 14. Créer la liste des instants t ainsi que le signal s
- échantillonné à $fe = 4\ kHz$;
- sur une durée $Tacq = 1s$;
- comportant une composante sinusoïdale de fréquence $f1 = 50\ Hz$ et d'amplitude $1\ V$ et une autre composante sinusoïdale de fréquence $f2 = 400\ Hz$ d'amplitude $0{,}5\ V$ correspondant à un bruit que l'on souhaite éliminer.
- on notera N le nombre de points et Te la période d'échantillonnage.

On pourra utiliser $np.arange(\text{start}, \text{stop}, \text{step})$ qui génère des valeurs dans l'intervalle $[start, stop)$ espacées de $step$.

➲ 15. Tracer ce signal sur une durée correspondant à 5 périodes $T1 = 1/f1$.

➲ 16. Générer $fourier$ la TFD de s à l'aide de la fonction $rfft$ de la bibliothèque $numpy.fft$:

➲ 17. Vérifier que le résultat $fourier$ est conforme aux informations présentes dans les notes ci-dessous extraites de la documentation de la fonction $numpy.fft.rfft$.

> When the DFT is computed for purely real input, the output is Hermitian-symmetric, i.e. the negative frequency terms are just the complex conjugates of the corresponding positive-frequency terms, and the negative-frequency terms are therefore redundant. This function does not compute the negative frequency terms, and the length of the transformed axis of the output is therefore `n//2 + 1`.
>
> When `A = rfft(a)` and fs is the sampling frequency, `A[0]` contains the zero-frequency term 0*fs, which is real due to Hermitian symmetry.
>
> If *n* is even, `A[-1]` contains the term representing both positive and negative Nyquist frequency (+fs/2 and -fs/2), and must also be purely real. If *n* is odd, there is no term at fs/2; `A[-1]` contains the largest positive frequency (fs/2*(n-1)/n), and is complex in the general case.

➲ 18. Justifier que pour N pair, les éléments du tableau $fourier$ correspondent à la liste de fréquences :

$$f_k = k \cdot \frac{1}{T_{acq}} \text{ pour } k \in \left[\!\left[0, \frac{N}{2}\right]\!\right]$$

➲ 19. Générer la liste des fréquences correspondant au spectre calculé à l'aide de $np.fft.rfftfreq(N, d = Te)$ avec N nombre de points et Te la période d'échantillonnage.

➲ 20. Tracer alors le spectre du signal : vérifier que le spectre obtenu est conforme à vos attentes et vérifier que la demi-largeur des raies est égale à $1/Tacq$.

➲ 21. Adapter le programme précédent pour obtenir le spectre d'un signal acquis avec LatisPro.
Réaliser l'acquisition avec des paramètres bien choisis puis exporter les courbes vers un fichier csv.
Ouvrir le fichier csv avec EXCEL, remplacer les « , » par des « . », si elle existe, supprimer la ligne qui donne le nom des colonnes. Enregistrer ce fichier sous le format txt avec séparateur tabulation.
Sous python :

- importer la bibliothèque os,
- utiliser la fonction $os.chdir(r"C: \dots")$ pour préciser le répertoire dans lequel se trouve le fichier de données,
- pour convertir le fichier texte de données en un tableau comportant 2 colonnes utiliser la fonction $np.loadtxt : t, U = np.loadtxt(" \dots .txt", unpack = True)$.

3) Retour sur le filtrage numérique – Analyse spectrale

La fonction de transfert d'un filtre passe-bas s'écrit sous forme canonique :

$$\underline{H} = \frac{H_0}{1 + j\omega\tau}$$

Dans la suite, on prendra $H_0 = 1$.

On considère un signal $e(t)$ échantillonné avec une période d'échantillonnage T_e.

✎➲ 22. Déterminer la relation de récurrence, issue du schéma numérique d'Euler explicite, permettant de calculer s_{n+1} à partir de $e_n = e(t_n = nT_e)$ et $s_n = s(t_n = nT_e)$.

Ainsi, on peut construire point par point le signal numérique s à partir de e.

On considère une tension d'entrée somme de deux tensions sinusoïdales :

$$e(t) = e_1(t) + e_2(t)$$
$$e_1(t) = 2.\cos(2\pi f_1 t)$$
$$e_2(t) = 0{,}5.\cos(2\pi f_2 t)$$

Avec $f_1 = 10\ Hz$ et $f_2 = 500\ Hz$

On souhaite isoler le signal de basse fréquence. Pour cela, on filtre numériquement.

➲ 23. Proposer des valeurs, en les justifiant, pour la fréquence de coupure f_c du filtre permettant d'obtenir un filtrage adéquat.

➲ 24. Compléter le programme Python *« TP4C_filtrage et rfft_à compléter »* disponible sur Cahier de Prépa.

➲ 25. Comparer le spectre du signal d'entrée à celui du signal de sortie du filtre.

E) Synthèse sur le choix des paramètres d'acquisition

➲ 26. Compléter le bilan ci-dessous.

Choix de la durée d'acquisition :

→ Pour un signal dont on connaît la période T :

Choix de la fréquence d'échantillonnage :

→ Pour un signal sinusoïdal de fréquence f :

Plus elle est élevée, meilleure sera l'image du signal mais cela peut poser problème (volume de stockage de données…).

→ Pour un signal quelconque dont le spectre est borné en fréquence par f_{max} :

La **condition de Nyquist-Shannon** doit être respectée :

Si le signal d'entrée et le dispositif d'acquisition ne permettent pas de le vérifier, la solution est :

Rq : Le traitement numérique peut imposer d'autres contraintes.

Choix du calibre :

Plus il est faible, meilleur sera le pas de quantification et donc la résolution (en volt) du signal mais s'il est trop faible il y aura saturation pour les valeurs élevées.

→ Il faut donc choisir le calibre C :

🗎 DOC 1 : Documentation fonction rfft de la bibliothèque numpy.fft

numpy.fft.rfft

`fft.rfft(a, n=None, axis=-1, norm=None)` [source]

Compute the one-dimensional discrete Fourier Transform for real input.

This function computes the one-dimensional *n*-point discrete Fourier Transform (DFT) of a real-valued array by means of an efficient algorithm called the Fast Fourier Transform (FFT).

Parameters: **a** : *array_like*
Input array

n : *int, optional*
Number of points along transformation axis in the input to use. If *n* is smaller than the length of the input, the input is cropped. If it is larger, the input is padded with zeros. If *n* is not given, the length of the input along the axis specified by *axis* is used.

axis : *int, optional*
Axis over which to compute the FFT. If not given, the last axis is used.

norm : *{"backward", "ortho", "forward"}, optional*

> ℹ *New in version 1.10.0.*

Normalization mode (see **numpy.fft**). Default is "backward". Indicates which direction of the forward/backward pair of transforms is scaled and with what normalization factor.

> ℹ *New in version 1.20.0:* The "backward", "forward" values were added.

Returns: **out** : *complex ndarray*
The truncated or zero-padded input, transformed along the axis indicated by *axis*, or the last one if *axis* is not specified. If *n* is even, the length of the transformed axis is `(n/2)+1`. If *n* is odd, the length is `(n+1)/2`.

Raises: **IndexError**
If *axis* is not a valid axis of *a*.

🗎 DOC 2 : Documentation fonction rfftfreq de la bibliothèque numpy.fft

numpy.fft.rfftfreq

`fft.rfftfreq(n, d=1.0)` [source]

Return the Discrete Fourier Transform sample frequencies (for usage with rfft, irfft).

The returned float array *f* contains the frequency bin centers in cycles per unit of the sample spacing (with zero at the start). For instance, if the sample spacing is in seconds, then the frequency unit is cycles/second.

Given a window length *n* and a sample spacing *d*:

```
f = [0, 1, ...,     n/2-1,     n/2] / (d*n)   if n is even
f = [0, 1, ..., (n-1)/2-1, (n-1)/2] / (d*n)   if n is odd
```

Unlike **fftfreq** (but like **scipy.fftpack.rfftfreq**) the Nyquist frequency component is considered to be positive.

Parameters: **n** : *int*
Window length.

d : *scalar, optional*
Sample spacing (inverse of the sampling rate). Defaults to 1.

Returns: **f** : *ndarray*
Array of length `n//2 + 1` containing the sample frequencies.

Examples

```
>>> signal = np.array([-2, 8, 6, 4, 1, 0, 3, 5, -3, 4],
dtype=float)
>>> fourier = np.fft.rfft(signal)
>>> n = signal.size
>>> sample_rate = 100
>>> freq = np.fft.fftfreq(n, d=1./sample_rate)
>>> freq
array([  0.,  10.,  20., ..., -30., -20., -10.])
>>> freq = np.fft.rfftfreq(n, d=1./sample_rate)
>>> freq
array([  0.,  10.,  20.,  30.,  40.,  50.])
```

Annexe : Principe de l'algorithme de J.W. Cooley et John Tukey

La TFD du signal numérique à N échantillons $s_E = [s_0, s_1, \ldots, s_{N-1}]$ est une liste de N nombres complexes $[\underline{c}_0, \underline{c}_1, \ldots, \underline{c}_{N-1}]$ définie par :

$$\forall k \in [\![0, N-1]\!], \quad \underline{c}_k = \sum_{j=0}^{N-1} s_j \exp\left(-i\frac{2\pi k j}{N}\right) \quad (\text{avec } i^2 = -1). \tag{5.1}$$

On vérifie facilement que $\underline{c}_0$ est réel et que :

$$\forall k \in [\![1, N-1]\!], \quad \underline{c}_{N-k} = \underline{c}_k^*. \tag{5.2}$$

Des relations de récurrence pour les TFD Dans la formule (5.1) on sépare les termes correspondant à j pair de ceux correspondant à j impair :

$$\begin{aligned}\underline{c}_k &= \sum_{j'=0}^{N/2-1} s_{2j'} \exp\left(-i\frac{2\pi k(2j')}{N}\right) + \sum_{j'=0}^{N/2-1} s_{2j'+1} \exp\left(-i\frac{2\pi k(2j'+1)}{N}\right) \\ &= \sum_{j'=0}^{N/2-1} s_{2j'} \exp\left(-i\frac{2\pi k j'}{N/2}\right) + \exp\left(-i\frac{2\pi k}{N}\right) \sum_{j'=0}^{N/2-1} s_{2j'+1} \exp\left(-i\frac{2\pi k j'}{N/2}\right)\end{aligned}$$

Dans cette expression on reconnaît deux autres TFD :

- si on pose $\underline{P}_k = \sum_{j'=0}^{N/2-1} s_{2j'} \exp\left(-i\frac{2\pi k j'}{N/2}\right)$, alors $[\underline{P}_0, \underline{P}_1, \ldots, \underline{P}_{N/2-1}]$ est la TFD du signal à $N/2$ échantillons $[s_0, s_2, \ldots, s_{N-4}, s_{N-2}]$;
- si on pose $\underline{I}_k = \sum_{j'=0}^{N/2-1} s_{2j'+1} \exp\left(-i\frac{2\pi k j'}{N/2}\right)$, alors $[\underline{I}_0, \underline{I}_1, \ldots, \underline{I}_{N/2-1}]$ est la TFD signal à $N/2$ échantillons $[s_1, s_3, \ldots, s_{N-3}, s_{N-1}]$.

Ainsi, en notant :

$$w = \exp\left(-i\frac{2\pi}{N}\right).$$

on peut dire que :

$$\text{pour } 0 \leq k < \frac{N}{2}, \quad \underline{c}_k = \underline{P}_k + w^k \underline{I}_k, \tag{5.3}$$

D'autre part, pour $0 \leq k < N/2$:

- d'après la propriété (5.2) de la TFD : $\underline{c}_{N/2+k} = \underline{c}^*_{N/2-k}$;
- d'après (5.3) : $\underline{c}^*_{N/2-k} = \underline{P}^*_{N/2-k} + \exp\left(i\dfrac{2\pi(N/2-k)}{N}\right)\underline{I}^*_{N/2-k}$;
- d'après la propriété (5.2) dans le cas de la TFD d'un signal à $N/2$ échantillons : $\underline{P}^*_{N/2-k} = \underline{P}_k$ et $\underline{I}^*_{N/2-k} = \underline{I}_k$.

Finalement :

$$\text{pour } 0 \leq k < \frac{N}{2}, \quad \underline{c}_{N/2+k} = \underline{P}_k - w^k \underline{I}_k \tag{5.4}$$

Les relations (5.3) et (5.4) ramènent le calcul d'une TFD d'un signal à N échantillons à deux calculs de TFD de signaux à $N/2$ échantillons

Algorithme pour le calcul de la TFD d'un signal à $N = 2^p$ échantillons :

1. si $N = 1$ (*i.e.* $p = 0$), le programme renvoie la liste $[s_0]$;
2. si $N > 1$ (*i.e.* $p > 0$), on fait deux appels récursifs pour calculer les TFD des signaux à $N/2 = 2^{p-1}$ échantillons constitués par les échantillons de rang pair et les échantillons de rang impair du signal original, puis on calcule la TFD du signal original par les formules (5.3) et (5.4).