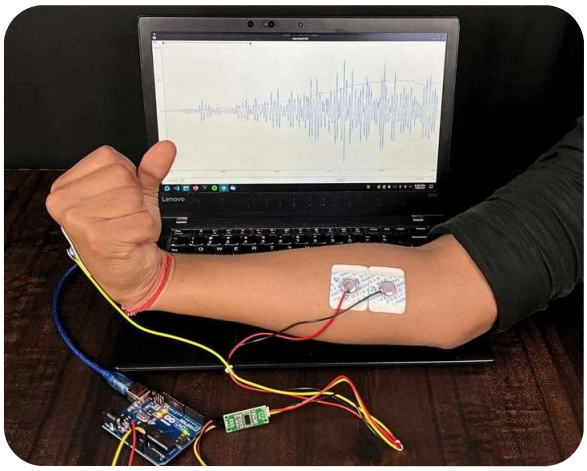
Filtrage numérique



L'électromyographie (EMG), est une technique de mesure pour évaluer l'activité électrique des muscles. Cette méthode implique l'enregistrement des signaux électriques produits par les cellules musculaires, les fibres musculaires, lorsqu'elles se contractent.

L'EMG permet de diagnostiquer certains troubles neuromusculaires ou d'évaluer les contractions musculaires. Cette technique est également utilisée pour le contrôle de certaines prothèses et orthèses.

Comme pour la plupart des mesures, le signal est initialement bruité et il est difficile de l'utiliser en l'état. Il est donc nécessaire de traiter ce signal, souvent grâce à des filtres.

Objectif :

L'objectif de ce TP est de présenter différentes méthodes de filtrage numérique. Nous verrons les méthodes de moyenne glissante, passe bas du premier ordre et du second ordre. Nous allons les programmer, les comparer et commenter leurs effets.

1. Présentation

On dispose d'une information numérique caractérisant un signal $u(t)$ bruité et on souhaite le filtrer afin d'éliminer les bruits. Nous allons donc programmer la moyenne glissante, différents filtres, et comparer leurs effets.

Le signal $u(t)$ est acquis avec une fréquence d'échantillonnage constante et on suppose avoir à notre disposition deux arrays de dimension N:

- $signal\_EMG$ array des valeurs échantillonnées de $u(t)$
- $temps$ array des valeurs des temps associés

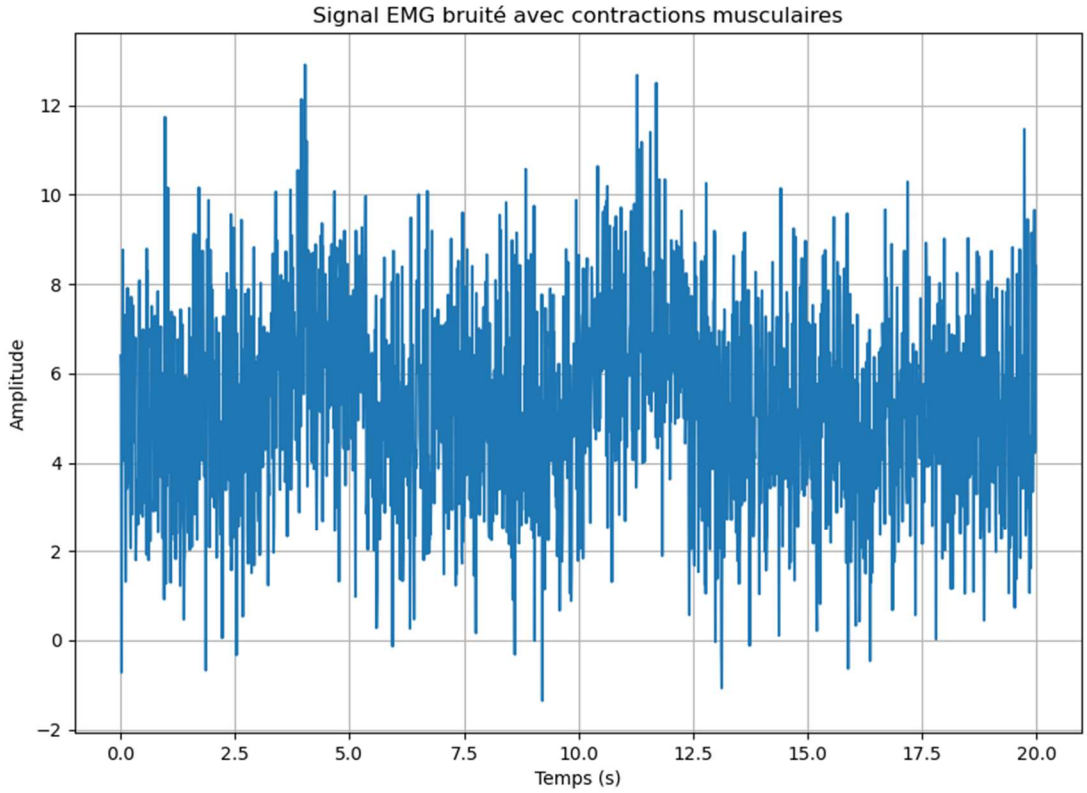
Ainsi :

$$u_i = signal\_EMG[i] = u(temps[i]), i \in [0, n]$$

Vous avez à disposition le script Python « Generation_EMG_bruite.py ». Voilà ce que génère son interprétation :

- Création d'un signal **aléatoire** bruité d'EMG contenant un certain nombre de contraction musculaire.
- Enregistre les deux arrays $signal\_EMG$ et $temps$ dans votre dossier de travail.
- Affiche le tracé du signal EMG bruité.

Voici un exemple de signal :



Q.1. **Exécuter ce script afin d'obtenir un signal similaire.**

Vous disposez également du script Python « TP_Filtrage_Numemique_etu.py » dans lequel se trouve un code à compléter pour le reste de ce TP.

Q.2. **Une bonne partie du code est déjà programmé, toutes les lignes ne sont pas à comprendre, cependant la logique algorithmique doit être comprise. Tout au long du TP n'hésitez pas prendre du temps pour comprendre le code qui entoure vos espaces « à compléter », cela vous aidera et vous fera progresser. Soyez curieux !**

2. Filtre par moyenne glissante

Le principe de cette méthode consiste à créer une liste $signal\_filtre$ dont chaque valeur $u'_i = signal\_filtre[i]$ vaut la moyenne de n valeurs prises autour de l'indice i (avant et/ou après). On appelle « ordre » ou « taille » de la moyenne glissante la valeur de n.

On choisit ici de prendre n valeurs centrées sur i. Voici comment calculer la liste les termes $u'_i \; \forall i \in [0, N-1]$ de $signal\_filtre$.

$si\ i < \lfloor n/2 \rfloor$	$si\ \lfloor n/2 \rfloor \leq i \leq N - \lfloor n/2 \rfloor$	$si\ i > N - \lfloor n/2 \rfloor$
Il n'existe pas $n/2$ termes jusqu'au terme d'indice i	Pas d'effet de bord	Il n'existe pas $n/2$ termes après le terme d'indice i
$u'_i = \frac{1}{\lfloor n/2 \rfloor + i + 1} \sum_{k=0}^{i+\lfloor n/2 \rfloor} u_k$	$u'_i = \sum_{k=i-\lfloor n/2 \rfloor}^{i+\lfloor n/2 \rfloor} \frac{u_k}{n}$	$u'_i = \frac{1}{\lfloor n/2 \rfloor - i + N} \sum_{k=i-\lfloor n/2 \rfloor}^{N} u_k$

Q.3. **Reproduite et compléter le tableau suivant pour n=3.**

i	0	1	2	3	4
u_i	-2	0	-1	2,5	3
u'_i	-1	-1	0.5	1,5	2,75

Q.4. **Compléter la fonction `moyenne_glissante(signal,fenetre)` prenant en argument l'array**

`signal`, l'ordre `fenetre` et renvoyant un array résultat du filtrage par moyenne glissante *`signal_filtre`*.

On rappelle qu'il est possible de créer un array de taille n remplit de 0 avec l'instruction `np.zeros(n)` (voir aussi np.zeros_like(array).

Q.5. **Vérifier le résultat sur l'exemple du tableau précédent.**

Q.6. **En utilisant la fonction `plot_comparison`, dont vous avez le prototype, tracer sur un graphique le signal bruité et le signal filtré pour n=20.**

Q.7. **En utilisant la fonction `plot_comparison_subplots`, dont vous avez le prototype, tracer sur une série de graphique le signal bruité et le signal filtré pour différente valeur de n, en rentrant n=20.**

Q.8. **Préciser l'influence de n sur la qualité du filtrage.**

3. Filtre numérique passe bas du premier ordre

On souhaite filtrer les mesures à l'aide d'un filtre passe bas du premier ordre. Cela revient à résoudre l'équation différentielle suivante :

$$\tau \frac{du_f(t)}{dt} + u_f(t) = u(t)$$

Avec τ la constante de temps du filtre, $u(t)$ est le signal à filtrer, $u_f(t)$ le signal filtré.

Il nous faut donc résoudre cette équation différentielle ! Une méthode simple et compatible avec le numérique (discrétisation) est la méthode d'Euler explicite. Cette méthode revient approximer la dérivée $\frac{du_f(t)}{dt}$ grâce à la formule du taux d'accroissement entre les valeurs $u_f(t)$ et $u_f(t-dt)$. On peut par exemple utiliser une différence finie arrière, ce qui donne $\frac{du_f(t)}{dt} = \frac{u_f(t)-u_f(t-\Delta t)}{\Delta t}$ avec Δt l'intervalle de temps entre deux échantillons successifs.

Q.9. **Donner l'expression de u_i^f en fonction u_{i-1}^f, u_i et Δt en utilisant la méthode d'Euler proposée.**

Q.10. **Montrer que cette expression peut se mettre sous la forme $u_i^f = u_{i-1}^f + \frac{\Delta t(u_i - u_{i-1}^f)}{\tau + \Delta t}$.**

Q.11. **Proposer une valeur pour u_0^f.**

Q.12. **Compléter la fonction `filtre_passe_bas_premier_ordre(signal, tau)` à partir du prototype de la fonction.**

Q.13. **En utilisant la fonction `plot_comparison`, dont vous avez le prototype, tracer sur un graphique le signal bruité et le signal filtré pour $\tau = 0.1$.**

Q.14. **En utilisant la fonction `plot_comparison_subplots`, dont vous avez le prototype, tracer sur une série de graphique le signal bruité et le signal filtré pour différente valeur de τ, en rentrant $\tau = 0.1$.**

Q.15. **Quelle est l'influence de τ sur la qualité du filtrage ? En déduire quelle est l'influence de la pulsation de coupure sur la qualité du filtrage ? De la fréquence de coupure f sur la qualité du filtrage ?**

4. Filtrage numérique passe bas du 2° ordre (pour les plus rapide)

Soit le filtrage du second ordre suivant :

$$\frac{1}{{\omega_0}^2}\frac{d^2u_f(t)}{dt^2}+\frac{2\xi}{\omega_0}\frac{du_f(t)}{dt}+u_f(t)=u(t)$$

Avec ω_0 la pulsation propre du filtre et ξ le coefficient d'amortissement.

Pour cette application, on remarquera qu'il est nécessaire que dt soit constant (ce qui est le cas du signal étudié) pour appliquer la dérivée seconde simplement.

On peut alors approximer la dérivée d'ordre 2 : $\frac{d^2u_f(t)}{dt^2}=\frac{u_f(t+\Delta t)-2u_f(t)+u_f(t-\Delta t)}{\Delta t^2}$

Q.16. **En utilisant l'approximation précédente pour la dérivée d'ordre 1 et l'approximation donnée pour la dérivée d'ordre 2, donner l'expression de u_i^f en fonction u_{i-1}^f, u_{i+1}^f, u_i, ξ, ω_0 et Δt.**

Pour l'initialisation on pose $\begin{cases} u_0^f = u_0 \\ u_{-1}^f = u_0 \end{cases}$

Q.17. **En définissant les coefficients a, b et c, montrer que l'équation de récurrence est**

```
signal_filtre[i + 1] = a * signal[i] - b * signal_filtre[i] + c * signal_filtre[i - 1]
```

Q.18. **L'initialisation de la première valeur sera telle suffisante ? Compléter le code de la fonction `filtre_passe_bas_second_ordre (signal, omega0, xi)`.**

Q.19. **On prendra xi=1, choisir une valeur de omega0 de manière à pouvoir comparer avec le filtre du premier ordre.**

Q.20. **En utilisant la fonction `plot_multiple_filter_comparison` tracer sur la même figure plusieurs résultats de filtrage en changeant des paramètres.**

Q.21. **Quelle est l'influence de omega0 sur le traitement ? Idem pour xi ?**

5. Bilan

Q.22. **Comparer les différents filtrages proposés (complexité, spécificités du filtrage obtenu, ...).**

Q.23. **Ajuster les paramètres pour avoir le plus proprement possible les contractions musculaires pour chaque filtre. Quel filtre semble le plus adapté ?**