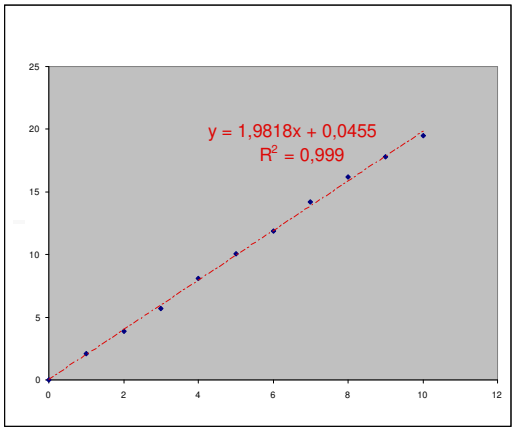
# 1. Rappel : approximation par les moindres carrés




*La pratique expérimentale conduit souvent à recherche d'une modélisation par une droite de la relation entre deux variables. Les divers outils informatiques à votre disposition (comme tout simplement votre calculatrice ou un tableur tel qu'Excel) possèdent des fonctions qui mettent en œuvre cette modélisation de manière automatique. L'objectif de ces quelques lignes est d'exposer les outils mathématiques qui en sont à l'origine et d'illustrer quelques applications sur Excel.*

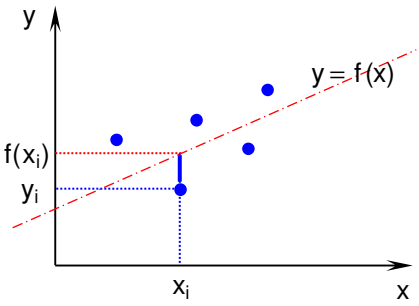
## I – FORMULATION DU PROBLÈME GÉNÉRAL

### I-1 Objectif

Soient n points expérimentaux à deux variables : $P_i = (x_i, y_i)$.

On cherche à approximer la distribution de ces points par une fonction donnant y = f(x), cette fonction (dont on fixe a priori le type) dépendant de k paramètres $p_k$. Par exemple on peut retenir a priori :

- l'approximation linéaire qui dépend de k=1 paramètre a : $y = ax$
- l'approximation par une loi de puissance qui dépend aussi de k=1 paramètre a : $y = x^a$
- l'approximation affine qui dépend de k=2 paramètres a et b : $y = ax + b$
- l'approximation polynomiale de degré 2 qui dépend de k=3 paramètres a, b et c : $y = ax^2 + bx + c$
- etc.




Pour chaque point, on caractérise l'erreur commise par l'approximation par : $e_i = y_i - f(x_i)$.

L'objectif consiste alors à minimiser cette erreur sur l'ensemble des n points.

### I-2 Méthode des moindres carrés.

Une première idée pourrait être de minimiser la somme de ces n erreurs. Mais cette somme n'est pas révélatrice de la dispersion : en effet, par exemple, des erreurs peuvent s'annuler deux à deux, ce qui offre une somme des erreurs nulles indépendamment de la dispersion. L'idée consiste alors à minimiser la somme de ces erreurs élevées au carré.

On construit alors la grandeur : $E = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - f(x_i))^2$

Puis on résout le système de k équations à k inconnues $\frac{\partial E}{\partial p_k} = 0$

Les k paramètres solutions de ce système de Cramer définissent alors la fonction d'approximation f(x).

## II – CAS DE L'APPROXIMATION LINÉAIRE

Fonction d'approximation : $y = ax$. Paramètre cherché : a.

### II-1 Mise en œuvre de la méthode des moindres carrés

Compte tenu de la méthode exposée, on construit :

$E = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - f(x_i))^2 = \sum_{i=1}^{n} (y_i - ax_i)^2 = \sum_{i=1}^{n} y_i^2 - 2ax_iy_i + a^2x_i^2$ , puis on calcule : $\frac{\partial E}{\partial a} = \frac{dE}{da} = \sum_{i=1}^{n} -2x_iy_i + 2ax_i^2$

qui s'annule pour : $a = \frac{\sum_{i=1}^{n} x_iy_i}{\sum_{i=1}^{n} x_i^2}$. Ce qui définit la fonction d'approximation : $\boxed{y = \frac{\sum_{i=1}^{n} x_iy_i}{\sum_{i=1}^{n} x_i^2} x}$

### II-2 Qualité de l'approximation

Si maintenant on cherche la fonction d'approximation $x = a'y$, on procède de la même manière :

$E = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (x_i - g(y_i))^2 = \sum_{i=1}^{n} (x_i - a'y_i)^2 = \sum_{i=1}^{n} x_i^2 - 2a'x_iy_i + a'^2y_i^2$ , d'où: $\frac{\partial E}{\partial a'} = \frac{dE}{da'} = \sum_{i=1}^{n} -2x_iy_i + 2a'y_i^2$

qui s'annule pour : $a' = \frac{\sum_{i=1}^{n} x_iy_i}{\sum_{i=1}^{n} y_i^2}$. Ce qui définit la fonction d'approximation : $x = \frac{\sum_{i=1}^{n} x_iy_i}{\sum_{i=1}^{n} y_i^2} y$

Si l'approximation est parfaite, c'est-à-dire si pour tous les points $y_i = ax_i$, on a alors, bien évidemment, des fonctions inverses qui se traduisent par $a.a' = 1$. Sinon les deux coefficients ne sont pas l'inverse l'un de l'autre.

On définit alors le nombre $R^2 = a.a'$, appelé coefficient de détermination. Plus ce coefficient est proche de 1, meilleure est l'approximation, il permet donc d'en apprécier la qualité.
On utilise parfois le nombre $R = \sqrt{R^2}$ affectée du signe de a. Il est appelé coefficient de corrélation.

Coefficient de détermination : $\boxed{R^2 = \frac{(\sum_{i=1}^{n} x_iy_i)^2}{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2}}$

## III – CAS DE L'APPROXIMATION AFFINE

Fonction d'approximation : $y = ax + b$. Paramètres cherchés : a et b

### III-1 Mise en œuvre de la méthode des moindres carrés

Compte tenu de la méthode exposée, on construit :

$$E = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - f(x_i))^2 = \sum_{i=1}^{n} (y_i - ax_i - b)^2 = \sum_{i=1}^{n} y_i^2 - 2ax_iy_i - 2by_i + a^2x_i^2 + 2abx_i + b^2$$

Puis on calcule : $\frac{\partial E}{\partial a} = \sum_{i=1}^{n} -2x_iy_i + 2ax_i^2 + 2bx_i$

et $\frac{\partial E}{\partial b} = \sum_{i=1}^{n} -2y_i + 2ax_i + 2b$

D'où le système de deux équations à deux inconnues a et b à résoudre :

$$\begin{cases} \sum_{i=1}^{n} -2x_iy_i + 2ax_i^2 + 2bx_i = 0 \\ \sum_{i=1}^{n} -2y_i + 2ax_i + 2b = 0 \end{cases}$$

Que l'on écrira sous la forme :

$$\begin{cases} -\sum_{i=1}^{n} x_iy_i + a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = 0 \\ -\sum_{i=1}^{n} y_i + a\sum_{i-1}^{n} x_i + nb = 0 \end{cases} \quad \text{ou encore} \quad \begin{cases} a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_iy_i \\ a\sum_{i=1}^{n} x_i + nb = \sum_{i=1}^{n} yi \end{cases}$$

dont la résolution fournit :

$$a = \frac{\begin{vmatrix} \sum_{i=1}^{n} x_i\,y_i & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} y_i & n \end{vmatrix}}{\begin{vmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{vmatrix}} = \frac{n\sum_{i=1}^{n} x_i\,y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} \quad \text{et} \quad b = \frac{\begin{vmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i\,y_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} y_i \end{vmatrix}}{\begin{vmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{vmatrix}} = \frac{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_iy_i}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

Ce qui définit la fonction d'approximation :

$$\boxed{y = \frac{n\sum_{i=1}^{n} x_i\,y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} x + \frac{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_iy_i}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}}$$

### III-2 Qualité de l'approximation

Si maintenant on cherche la fonction d'approximation $x = a'y + b'$, on procède de la même manière pour aboutir à (il suffit de changer les $x_i$ en $y_i$ dans l'écriture ci-dessus) :

$$y = \frac{n\sum_{i=1}^{n} x_i\,y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2} x + \frac{\sum_{i=1}^{n} y_i^2 \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_iy_i}{n\sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2}$$

Comme pour l'approximation linéaire, on définit alors un coefficient de détermination qui est le produit des deux pentes et qui sera d'autant plus proche de 1 que l'approximation sera bonne :

$$R^2 = \frac{[n\sum_{i=1}^{n} x_i\, y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i\,]^2}{[n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i\,)^2].[n\sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i\,)^2]}$$

# 2. Dérivation et intégration numériques

## 1. Introduction et rappels

L'intégration et la dérivation forment le cœur de l'analyse mathématique. Il n'est pas toujours possible de calculer la dérivée et la primitive analytique d'une fonction, pour diverses raisons :

- on ne connaît les primitives que d'un très petit nombre de fonctions,
- ces primitives ne sont pas toujours calculables analytiquement,
- l'expression analytique de la fonction elle-même n'est pas toujours connue (expression numérique issue d'un capteur par exemple).

Nous rapellons ici la formule de Taylor qui démontre qu'une fonction n+1 fois dérivable au voisinage d'un point $x_0$ peut être approximée par une fonction polynômiale dont les coefficients dépendent uniquement des dérivées de la fonction en ce point.

$$f(x)=f(x_0)+\frac{x-x_0}{1!}\frac{df}{dx}(x_0)+\frac{(x-x_0)^2}{2!}\frac{d^2f}{dx^2}(x_0)+...+\frac{(x-x_0)^n}{n!}\frac{d^nf}{dx^n}(x_0)+R_{n+1}(x)$$

avec

$$R_{n+1}=\frac{(x-x_0)^{n+1}}{(n+1)!}\frac{d^{n+1}f}{dx^{n+1}}(\xi) \quad \text{avec} \quad \xi\in[x;x_0]$$

## 2. Dérivée numérique

Nous allons maintenant étudier des méthodes numériques pour le calcul approché de

$$\frac{df}{dx}(x)$$

Le procédé le plus simple s'inspire de la définition même de la dérivée:

$$\frac{df}{dx}(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

D'après le théorème de Taylor, nous pouvons écrire précisement:

$$\frac{df}{dx}(x)=\frac{f(x+h)-f(x)}{h}-\frac{h}{2}\frac{d^2f}{dx^2}(\xi) \quad \text{avec} \quad \xi\in[x;x+h]$$

Le terme d'erreur $\frac{-h}{2}\frac{d^2f}{dx^2}(\xi)$ est proportionnel à h et la formule est exacte pour une fonction linéaire.

On peut améliorer ce résultat sans coût suplémentaire en s'appuyant sur le dévelloppement de Taylor d'ordre 3 des fonctions $f(x+h)$ et $f(x-h)$ :

$$f(x+h)=f(x)+h\frac{df}{dx}(x)+\frac{h^2}{2}\frac{d^2f}{dx^2}(x)+\frac{h^3}{6}\frac{d^3f}{dx^3}(\xi_1) \quad \text{avec} \quad \xi_1\in[x;x+h]$$

$$f(x-h)=f(x)-h\frac{df}{dx}(x)+\frac{h^2}{2}\frac{d^2f}{dx^2}(x)-\frac{h^3}{6}\frac{d^3f}{dx^3}(\xi_2) \quad \text{avec} \quad \xi_2\in[x-h;x]$$

Les termes en $h^2$ disparaissent lors de la différence terme à terme des deux lignes :

$$\frac{df}{dx}(x)=\frac{f(x+h)-f(x-h)}{2h}-\frac{h^2}{6}\frac{d^3f}{dx^3}(\xi)$$

Cette fois-ci, l'erreur de troncature est maintenant en $h^2$ et la formule est exacte pour les polynomes du second degré.

Nous retiendrons trois façons d'approximer la dérivée d'une fonction:

- Différence finie progressive: $\frac{df}{dx}(x)\approx\frac{f(x+h)-f(x)}{h}$
- Différence finie retrograde: $\frac{df}{dx}(x)\approx\frac{f(x)-f(x-h)}{h}$
- Différence finie centrée: $\frac{df}{dx}(x)\approx\frac{f(x+h)-f(x-h)}{2h}$

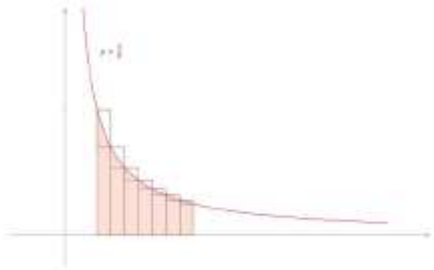
La démonstration précédente a permis de mettre en évidence que la méthode de la différence finie centrée donnait de meilleurs résultats sans pour autant nécessiter d'évaluation supplémentaire de la fonction $f$ .

## *3. Intégration numérique*

Nous allons maintenant étudier des méthodes numériques pour le calcul approché de

$$\int_a^b f(x)dx$$

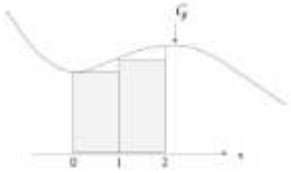
L'idée générale consiste à diviser l'intervalle d'intégration et à estimer la surface délimitée par la courbe et l'axe des abscisses.



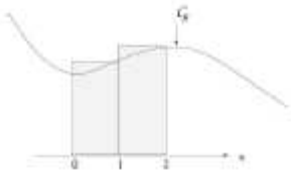
$$\int_a^b f(x)dx=\sum_{i=0}^{n}\int_{a+ih}^{a+(i+1)h} f(x)dx \quad \text{avec} \quad h=\frac{b-a}{n}$$

## 3.1 Méthodes élémentaires

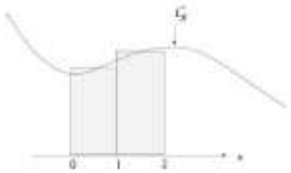
La première approche consiste à évaluer chaque surface en calculant la somme des volumes formés par des rectangles.



Méthode des rectangles à droite



Méthode des rectangles à gauche



Méthode du point milieu

Méthode des rectangles à gauche: $\int_a^{a+h} f(x)dx \approx h.f(a)$

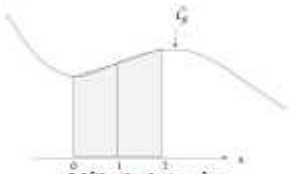
Méthode des rectangles à droite: $\int_a^{a+h} f(x)dx \approx h.f(a+h)$

Méthode du point milieu: $\int_a^{a+h} f(x)dx \approx h.f\left(a+\frac{h}{2}\right)$

En utilisant le théorème de Taylor, il est possible de démontrer que la méthode du point milieu minimise l'erreur.

## 3.2 Méthodes du trapèze

La méthode du trapèze consiste à approximer l'aire par un trapéze épousant la courbe aux points ayant pour abscisses $a$ et $a+h$ .

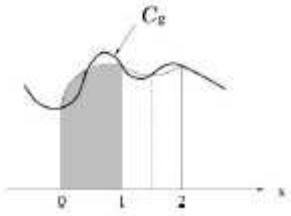


Méthode du trapèze

Méthode du trapèze: $\int_a^{a+h} f(x)dx \approx \frac{h}{2}(f(a)+f(a+h))$

## 3.3 Méthodes de Simpson

La méthode de Simpson consiste à interpoler la courbe par un polynome de degré 2 aux points ayant pour abscisses $a$ , $a+\frac{h}{2}$ et $a+h$ .




Méthode de Simpson

Formule de quadrature de Simpson: $\int_a^{a+h} f(x)dx \approx \frac{h}{6}\left[f(a)+4f\left(a+\frac{h}{2}\right)+f(a+h)\right]$

## 3.4 Influence du bruit de mesure en dérivation numérique

Lorsque la courbe est issue d'une mesure, elle est généralement entachée d'un léger bruit, qui peut devenir catastrophique pour l'évaluation de la dérivée (FIG 14). En effet, si les points de mesure restent "en moyenne" au voisinage de la valeur mesurée, il existe des fluctuations rapides (c'est-à-dire à la fréquence d'échantillonnage) entre les points successifs (voir zoom de la FIG 14).
Le calcul de la dérivée conduit à déterminer la pente entre deux points successifs, ce qui perturbe fortement le signal dérivé et cache les évolutions lentes du signal (lentes devant la période d'échantillonnage).

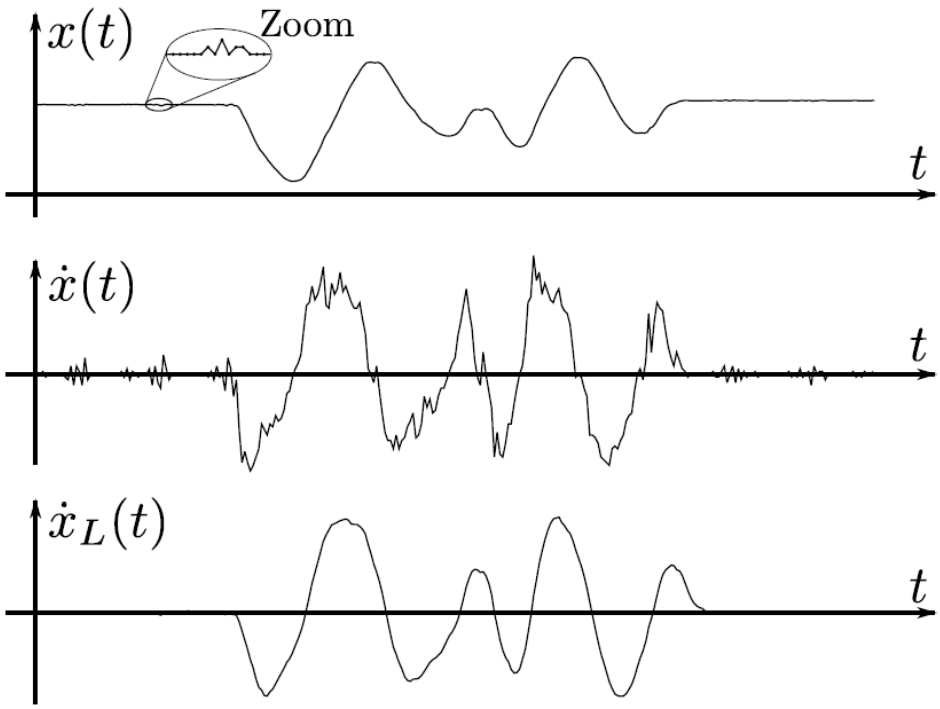



FIGURE 14 – Mesure d'une position au cours du temps $x(t)$ par un capteur potentiométrique, dérivée à 1 pas $\dot{x}(t)$ et dérivée à 1 pas lissée en effectuant la différence sur 10 pas.

Deux solutions sont possibles :
• filtrer (ou lisser) le signal d'origine pour supprimer l'essentiel du bruit, puis dériver,
• calculer la dérivée sur un temps plus long que le temps d'échantillonnage, par exemple pour une méthode à 1 pas en calculant la pente entre deux points espacés de $k$ pas (solution adoptée pour $\dot{x}L(t)$ sur la FIG 14, avec $k = 10$).

Dans les deux cas, le signal dérivé sera entaché d'un retard sur le signal d'origine, ce qui oblige à trouver un compromis entre la qualité du signal dérivé et le retard. $k$ est généralement de l'ordre de 5 à 20 pas selon le bruit, le pas d'échantillonnage, les fréquences à observer dans le signal et le retard admissible.

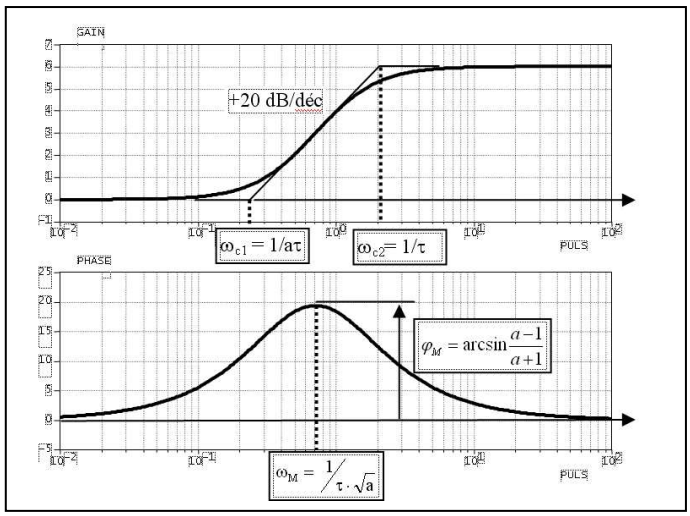
Pour un lissage par moyenne mobile, on peut montrer que les deux méthodes s'avèrent identiques.

## 3. Exemple d'application - Numérisation d'un correcteur à avance de phase

Son équation dans le domaine de Laplace est :

S(p) = $\dfrac{1+a.\tau.p}{1+\tau.p}$ . E(p) avec a > 1

τ : constante de temps ;
a : coefficient d'avance de phase.




Modifions cette égalité pour ne faire apparaître que des p au numérateur :

$$S(p).(1+\tau.p) = (1+a.\tau.p).E(p)$$

ou encore :

$$S(p) + \tau.p.S(p) = E(p) + a.\tau.p.E(p)$$

Repassons dans le domaine temporel et
Remplaçons la multiplication par p du domaine de Laplace par la dérivation temporelle :

$$S(t) + \tau.\frac{dS(t)}{dt} = E(t) + a.\tau.\frac{dE(t)}{dt}$$

Intégrons cette égalité entre les deux instants d'échantillonnage : (n-1)Te et nTe :

$$\int_{(n-1)Te}^{nTe} S(t).dt + \tau.\left[S(nTe) - S((n-1)Te)\right] = \int_{(n-1)Te}^{nTe} E(t).dt + a.\tau.\left[E(nTe) - E((n-1)Te)\right]$$

Soit en introduisant les notations discrètes :

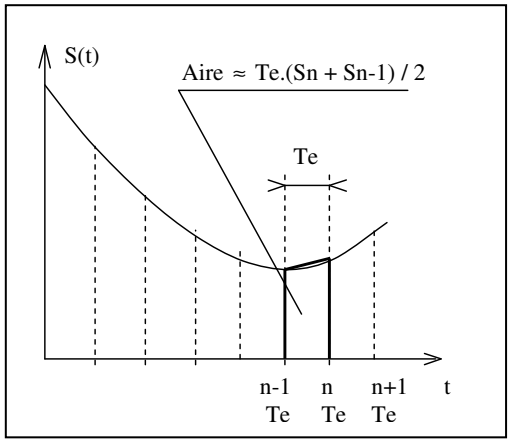
$$\int_{(n-1)Te}^{nTe} S(t).dt + \tau.(S_n - S_{n-1}) = \int_{(n-1)Te}^{nTe} E(t).dt + a.\tau.(E_n - E_{n-1})$$

On exprime les termes intégraux par la méthode des trapèzes :

$$\int_{(n-1)Te}^{nTe} S(t).dt = aire\_sous\_courbe \approx \frac{Te}{2}(S_n + S_{n-1})$$

De même :

$$\int_{(n-1)Te}^{nTe} E(t).dt = aire\_sous\_courbe \approx \frac{Te}{2}(E_n + E_{n-1})$$




Et on obtient :

$$\frac{Te}{2}(S_n + S_{n-1}) + \tau.(S_n - S_{n-1}) = \frac{Te}{2}(E_n + E_{n-1}) + a.\tau.(E_n - E_{n-1})$$

$$S_n(\frac{Te}{2} + \tau) + S_{n-1}(\frac{Te}{2} - \tau) = E_n.(\frac{Te}{2} + a.\tau) + E_{n-1}.(\frac{Te}{2} - a.\tau)$$

ce qui permet d'exprimer le terme Sn cherché :

$$S_n(avance\ de\ phase) = \frac{2}{Te + 2.\tau}.\left[(\tau - \frac{Te}{2}).S_{n-1} + (\frac{Te}{2} + a.\tau).E_n + (\frac{Te}{2} - a.\tau).E_{n-1}\right]$$

**Nota** : si on utilisait la méthode des rectangles avec $\int_{(n-1)Te}^{nTe} S(t).dt \approx Te.S_n$ et $\int_{(n-1)Te}^{nTe} E(t).dt \approx Te.E_n$

On obtiendrait : $Te.S_n + \tau.(S_n - S_{n-1}) = Te.E_n + a.\tau.(E_n - E_{n-1})$

Alors : $S_n.(Te + \tau) - \tau.S_{n-1} = E_n(Te + a.\tau) - a.\tau.E_{n-1}$

Et donc : $S_n = \dfrac{1}{Te + \tau}\left[\tau.S_{n-1} + (Te + a.\tau).E_n - a.\tau.E_{n-1}\right]$

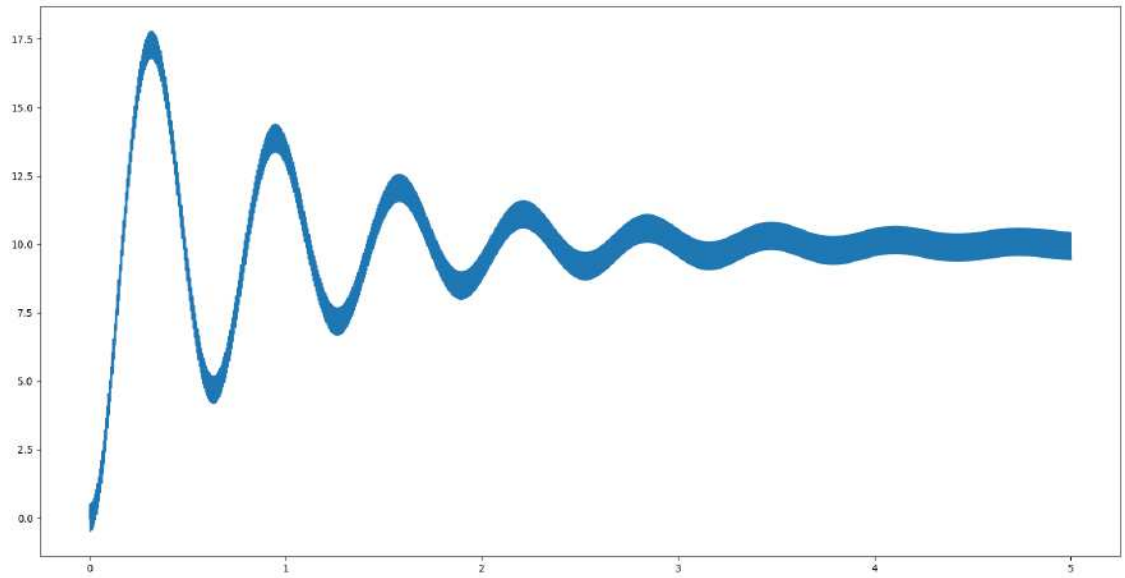
## 4. Exemple d'application – Filtrage numérique

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 à une dimension $N$ (on pourra les traiter comme des listes) :

- $U$ array des valeurs échantuillonnées de $u(t)$
- $T$ array des valeurs des temps associés

Ainsi :

$$u_i = U[i] = u(T[i]), i \in [0, n]$$




Ce signal correspond à la réponse théorique d'un système du second ordre à un échelon, bruité par un signal sinusoïdal (lisez le code pour comprendre comment il est réalisé).

L'objectif est de filtrer le bruit sur ce signal.

### *Filtrage par moyenne glissante*

Le principe de cette méthode consiste à créer une liste $U'$ dont chaque valeur $u'_i = U'[i]$ vaut la moyenne de $n$ valeurs prises proche de l'indice $i$ (avant, après et/ou autour). On appelle « ordre de la moyenne glissante » la valeur de $n$.
On choisit ici de prendre $n$ valeurs avant $i$. Voici comment calculer la liste les termes $u'_i \; \forall i \in [0, N-1]$ de $U'$.

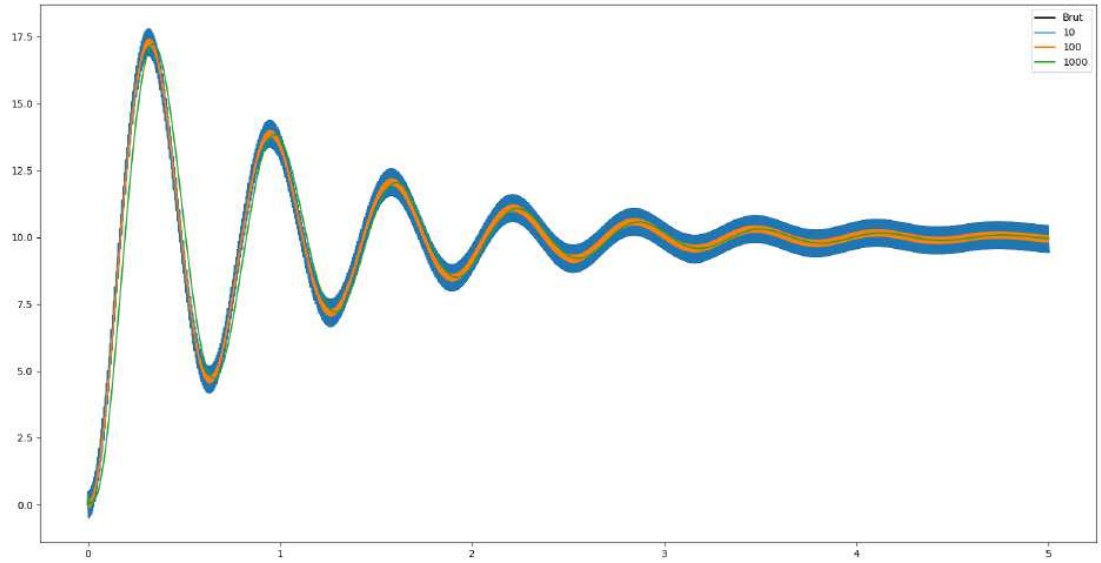
| $si\ i < n-1$ | $si\ i \geq n-1$ |
|---|---|
| Il n'existe pas $n$ termes jusqu'au terme d'indice $i$ | Il existe $n$ termes jusqu'à $i$ |
| $u'_i = \sum_{k=0}^{i} \frac{u_k}{i+1}$ | $u'_i = \sum_{k=i-n+1}^{i} \frac{u_k}{n}$ |

| $i$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $u_i$ | $-2$ | 0 | $-1$ | 2,5 | 3 |
| $u'_i$ | $-2$ | $-1$ | $-1$ | 0,5 | 1,5 |

**Question 2: Créer la fonction Filtre_MG(U,n) prenant en argument l'array U, l'ordre n et renvoyant un array résultat du filtrage par moyenne glissante de U d'ordre n**

**Question 3: Vérifier le résultat sur l'exemple du tableau précédent**

**Question 4: Tracer sur un même graphique la courbe des mesures brutes et celle du filtrage de U pour n=10, 100 et 1000**




**Question 5: Préciser l'influence de n sur la qualité du filtrage**

Si n trop faible, bruit de mesure atténué, non supprimé

Si n trop grand, toute la courbe est lissée, faisant disparaître les oscillations du signal avec un léger retard

```
33
34 def Filtre_MG(U,n): # Traduction littÃ©rale des formules
35     N = len(U)
36     Res = np.zeros(N)
37     for i in range(N):
38         upi = 0
39         if i < n-1:
40             for k in range(i+1):
41                 upi += U[k]
42             upi /= (i+1)
43         else:
44             for k in range(i-n+1,i+1):
45                 upi += U[k]
46             upi /= n
47         Res[i] = upi
48     return Res
```

# *Filtrage numérique passe bas du 1° ordre*

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

$$\tau\frac{du'(t)}{dt}+u'(t)=u(t) \quad ; \quad H(p)=\frac{1}{1+\tau p}$$

Avec $f=\frac{1}{2\pi\tau}$ la fréquence de coupure du filtre.

**Question 6: Donner l'expression de $u'_{i+1}$ $\forall i \in [1,N]$ en fonction de $u_i$, $u'_i$, $t_i$, $t_{i+1}$ et $\tau$ en utilisant la méthode d'Euler explicite**

---

$$\tau\frac{du'(t)}{dt}+u'(t)=u(t)$$

On approche la dérivée de la manière suivante :

$$\frac{du'(t)}{dt}=\frac{u'(t+dt)-u'(t)}{dt}$$

On a alors :

$$\tau\frac{u'(t+dt)-u'(t)}{dt}+u'(t)=u(t)$$

$$\tau u'(t+dt)-\tau u'(t)+dtu'(t)=dtu(t)$$

$$\tau u'(t+dt)=\tau u'(t)-dtu'(t)+dtu(t)$$

$$u'(t+dt)=u'(t)+\frac{dt}{\tau}\left(u(t)-u'(t)\right)$$

Soit avec les notations demandées :

$$u'_{i+1}=u'_i+\frac{(t_{i+1}-t_i)(u_i-u'_i)}{\tau}$$

---

```
103
104 def Filtre_PB1(U,T,tau):
105     N = len(U)
106     Up = np.zeros(N)
107     Up[0] = U[0]
108     for i in range(N-1):
109         ui = U[i]
110         upi = Up[i]
111         dt = T[i+1] - T[i]
112         du = ui - upi
113         upip1 = upi + dt*du/tau
114         Up[i+1] = upip1
115     return Up
116
```

# *Filtrage numérique passe bas du 2° ordre*

Soit le filtrage du second ordre suivant :

$$\frac{1}{{\omega_0}^2}\frac{d^2u'(t)}{dt^2}+\frac{2z}{\omega_0}\frac{du'(t)}{dt}+u'(t)=u(t) \quad ; \quad H(p)=\frac{1}{1+\frac{2z}{\omega_0}p+\frac{1}{{\omega_0}^2}p^2}$$

Avec $f_0=\frac{\omega_0}{2\pi\tau}$ la fréquence propre du filtre et $z$ 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.

**Question 11: En suivant la même démarche que pour le filtre du 1° ordre, réaliser le filtrage des données avec un passe bas du second ordre**

$$\frac{1}{{\omega_0}^2}\frac{d^2u'(t)}{dt^2}+\frac{2z}{\omega_0}\frac{du'(t)}{dt}+u'(t)=u(t)$$

On approche les dérivées de la manière suivante :

$$\frac{du'(t)}{dt}=\frac{u'(t+dt)-u'(t)}{dt}$$

$$\frac{d^2u'(t)}{dt^2}=\frac{u'(t+dt)-2u'(t)+u'(t-dt)}{dt^2}$$

On remplace dans l'équation :

$$\frac{1}{{\omega_0}^2}\frac{u'(t+dt)-2u'(t)+u'(t-dt)}{dt^2}+\frac{2z}{\omega_0}\frac{u'(t+dt)-u'(t)}{dt}+u'(t)=u(t)$$

$$\frac{1}{{\omega_0}^2dt^2}\big(u'(t+dt)-2u'(t)+u'(t-dt)\big)+\frac{2z}{\omega_0 dt}\big(u'(t+dt)-u'(t)\big)+u'(t)=u(t)$$

Posons :

$$\begin{cases} a = \dfrac{1}{{\omega_0}^2 dt^2} \\ b = \dfrac{2z}{\omega_0 dt} \end{cases}$$

$$a\big(u'(t+dt) - 2u'(t) + u'(t-dt)\big) + b\big(u'(t+dt) - u'(t)\big) + u'(t) = u(t)$$

$$au'(t+dt) - 2au'(t) + au'(t-dt) + bu'(t+dt) - bu'(t) + u'(t) = u(t)$$

$$(a+b)u'(t+dt) + (1-2a-b)u'(t) + au'(t-dt) = u(t)$$

$$u'(t+dt) = \frac{u(t) + (2a+b-1)u'(t) - au'(t-dt)}{a+b}$$

$$\boxed{u'_{i+1} = \frac{u_i + (2a+b-1)u'_i - au'_{i-1}}{a+b}}$$

Avec :

$$\begin{cases} 2a+b-1 = \dfrac{2}{{\omega_0}^2 dt^2} + \dfrac{2z}{\omega_0 dt} - 1 = \dfrac{2 + 2z\omega_0 dt - {\omega_0}^2 dt^2}{{\omega_0}^2 dt^2} \\ a+b = \dfrac{1}{{\omega_0}^2 dt^2} + \dfrac{2z}{\omega_0 dt} = \dfrac{1 + 2z\omega_0 dt}{{\omega_0}^2 dt^2} \end{cases}$$

Voulant que le signal filtré $u'$ présente la même dérivée nulle que le signal $u$, on pose :

$$\begin{cases} u'_0 = u_0 \\ u'_{-1} = u_0 \end{cases}$$

```
147
148 def Filtre_PB2(U,T,z,w0):
149     N = len(U)
150     Up = np.zeros(N)
151     Up[0] = U[0]
152     for i in range(N-1):
153         if i==0:
154             upim1 = Up[0]
155             # upim1 = Up[0] - (Up[1]-Up[0])
156         else:
157             upim1 = Up[i-1]
158         ui = U[i]
159         upi = Up[i]
160         dt = T[i+1] - T[i]
161         a = 1/((w0*dt)**2)
162         b = 2*z/(w0*dt)
163         upip1 = (ui + (2*a+b-1)*upi - a*upim1) / (a+b)
164         Up[i+1] = upip1
165     return Up
166
```

# 5. Exemple d'application - Extrait CCINP MP 2022 – Exolift, système d'aide à la montée d'échelle au sein d'un parc éolien.

### Présentation

Dans une problématique d'entretien du parc éolien, les techniciens sont appelés à monter et descendre plusieurs fois par jour des échelles pouvant mesurer jusqu'à 80 mètres. L'entreprise française Fixator, fabricant de treuils et de plateformes suspendues dans le domaine du bâtiment et des travaux publiques (BTP) depuis plus de 90 ans, a mis à profit son expérience pour concevoir un système autonome d'aide à la montée, Exolift. Ce système portatif sur batterie fonctionne indépendamment de l'éolienne (si le parc éolien est mis hors tension pour des raisons de sécurité, les techniciens de maintenance peuvent profiter pleinement de leur aide à la montée, contrairement aux systèmes installés à demeure sur l'échelle de l'éolienne). Exolift permet ainsi de réduire les temps d'arrêt des éoliennes. Exolift est également un équipement qui minimise l'investissement initial pour les exploitants des parcs éoliens car il s'utilise sur une simple sangle installée à demeure sur l'échelle, réduisant le temps d'installation et le coût matériel.

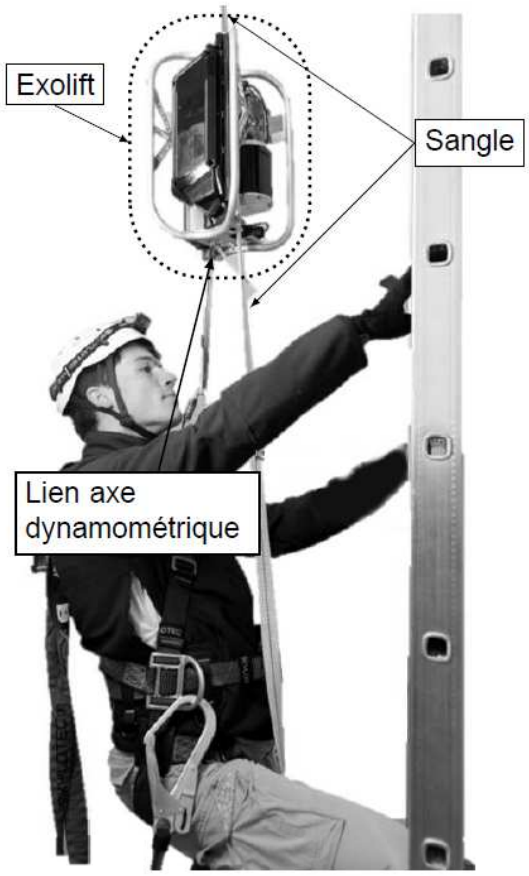



**Figure 1 -** Exolift utilisé sur une échelle

### Analyse structurelle

D'une manière générale, l'Exolift utilise le phénomène d'adhérence entre la sangle et un galet motorisé. La sangle étant fixée à l'échelle, lorsque le galet va tourner, l'ensemble Exolift va alors se déplacer le long de la sangle et supporter une partie du poids de l'utilisateur à la montée, comme à la descente (**figure 1**).

Il est principalement constitué (**figure 2**) d'un cadre 1, d'un panneau de commande 2, d'une batterie 3, d'un moteur électrique 4, d'un réducteur avec renvoi d'angle 5 et d'un galet motorisé 6. Le panneau de contrôle/commande permet à l'utilisateur d'obtenir des informations sur l'état du système (par le biais de 5 leds multicolores) et d'envoyer des ordres de commande (marche/arrêt via un interrupteur, monter/descendre via une manette de commande, arrêt d'urgence...). L'Exolift comporte également un axe dynamométrique 7, permettant de mesurer l'action mécanique de l'utilisateur sur l'Exolift, et un capteur à effet Hall au niveau de l'arbre du moteur permettant de mesurer la vitesse angulaire du moteur (uniquement pour des questions de sécurité). Un capteur de fin de course est positionné aux extrémités de l'échelle afin de stopper l'Exolift. Enfin, une option sur l'Exolift permet de le renvoyer à vide en haut ou en bas de l'échelle afin qu'une autre personne puisse l'utiliser. Toutes ces commandes et informations sont traitées par une carte électronique embarquée qui se charge de générer la commande du moteur tout en assurant la sécurité de l'utilisateur.

Le diagramme de définition des blocs (**figure 3**) présente les différents sous-systèmes et composants de l'Exolift. Le diagramme de blocs interne (**figure 4**) présente l'architecture de l'Exolift ainsi que les flux échangés entre les différents sous-systèmes. Un extrait du cahier des charges est présenté par le diagramme des exigences (voir **D6** du **Document Réponse**).

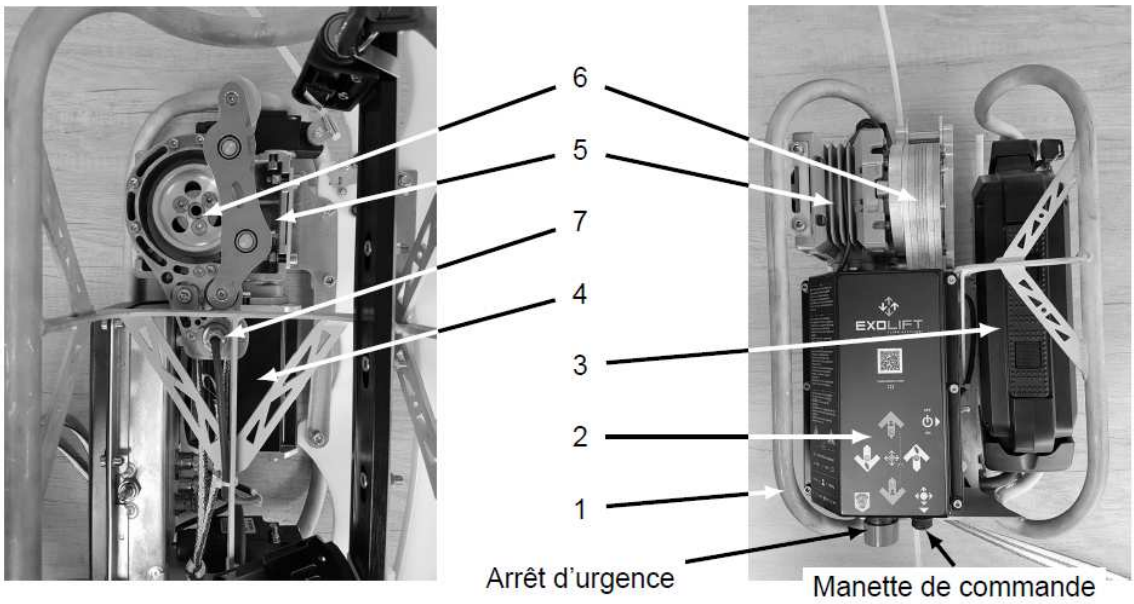



**Figure 2 -** À gauche, vue de l'Exolift suivant l'axe du galet (sans la batterie) ; à droite, vue de face avec la batterie

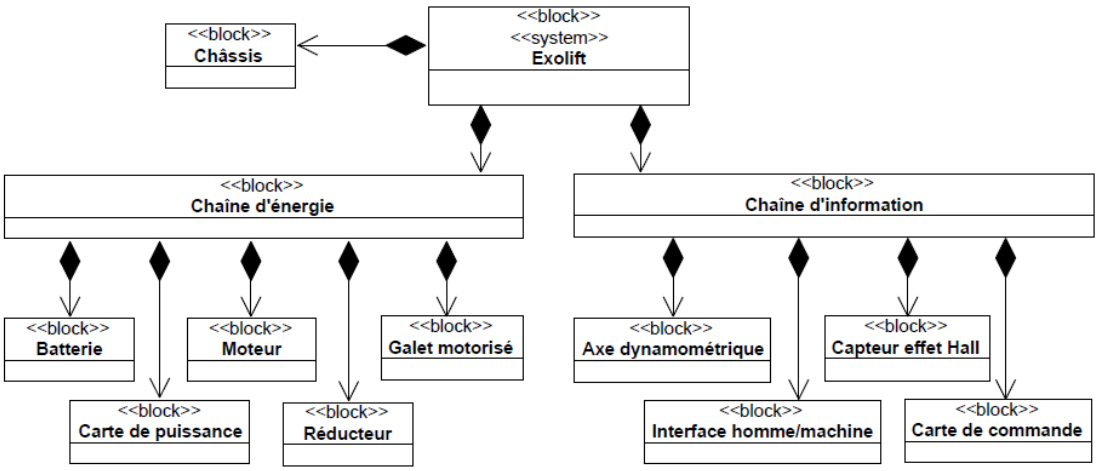



**Figure 3 -** Diagramme de définition des blocs partiel de l'Exolift

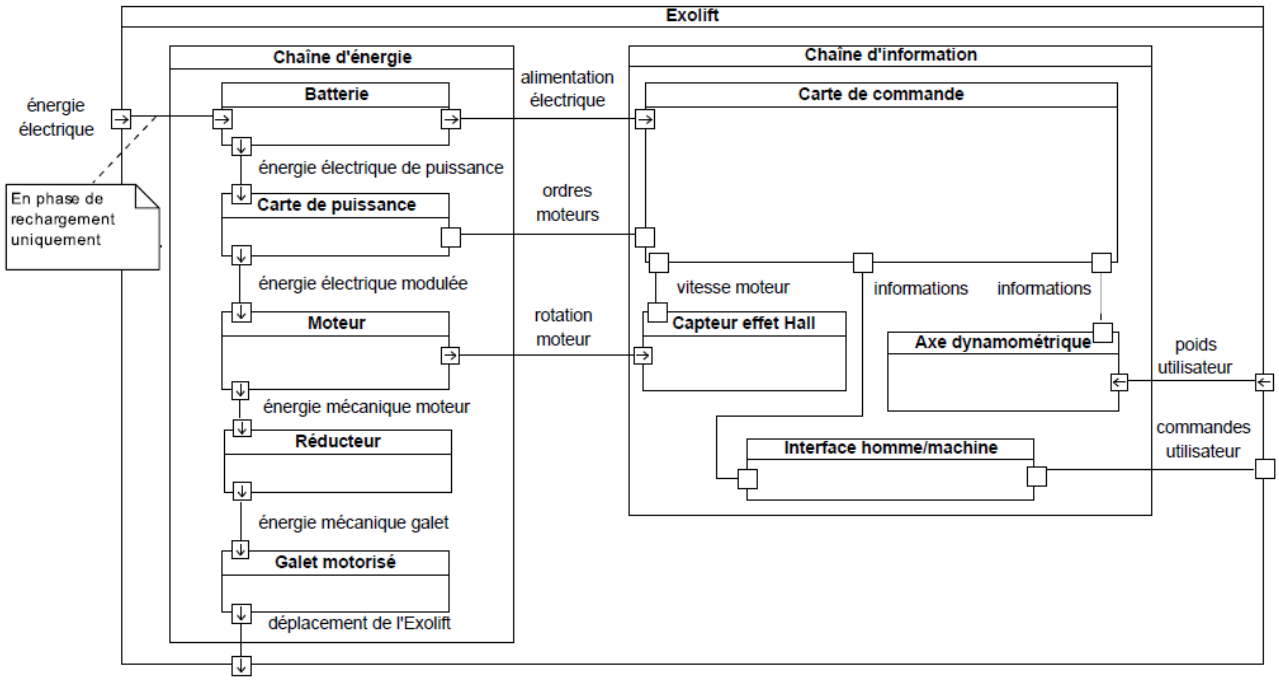



**Figure 4 -** Diagramme de blocs interne partiel de l'Exolift

**Q1.** Compléter la chaîne structurelle de l'Exolift du **Document Réponse 1** en précisant le nom des composants et la nature des flux échangés (I pour information, E pour énergie et M pour matière).

# Partie II - Étude de la chaîne d'information (*Informatique Pour Tous*)

***Objectif** : mettre en place le traitement des informations reçues de l'axe dynamométrique afin de les rendre utilisables par la carte de commande.*

L'effort de l'utilisateur sur l'Exolift $F_m$, permettant d'imposer la commande de vitesse de l'Exolift, est mesuré à l'aide d'un axe dynamométrique. Cette action mécanique déforme l'axe dynanométrique. Cette déformation est mesurée à l'aide de deux jauges d'extensiométrie (jauges de déformation) collées sur un arbre. Le signal délivré par l'axe dynanométrique est une tension liée à l'effort recherché.

Dans cette partie, l'étalonnage du capteur, puis le filtrage de sa sortie seront analysés et, ensuite, les différentes données enregistrées lors d'un étalonnage seront exploitées.

## II.1 - Étalonnage de l'axe dynanométrique

***Objectif** : déterminer le modèle de comportement du capteur afin de déterminer la relation entre la tension sortie du capteur et l'effort $F_m$ recherché (exigence 1.3.3.1.1).*

Un étalonnage, réalisé sur le site de production, permet d'obtenir la relation entre l'effort $F_m$ et le signal de sortie du capteur. La **figure 8** présente les résultats d'un étalonnage réalisé sur l'axe dynanométrique de l'Exolift. Ce graphique montre que la relation entre la tension de sortie du capteur et l'effort mesuré peut être approximée par une loi affine, qui sera considérée valide sur toute la plage d'utilisation de l'axe dynamométrique. Il est donc proposé ici de programmer une régression linéaire.

Pour réaliser la régression linéaire, la méthode des moindres carrés sera utilisée. Elle consiste à modéliser au mieux les points expérimentaux par une droite (**figure 9**).

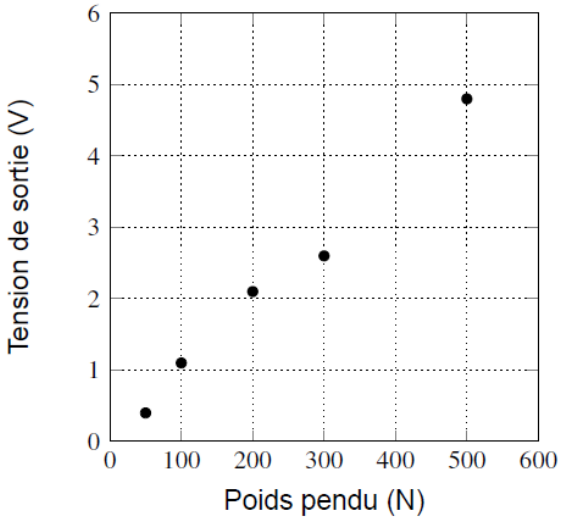



**Figure 8 -** Résultat d'un étalonnage

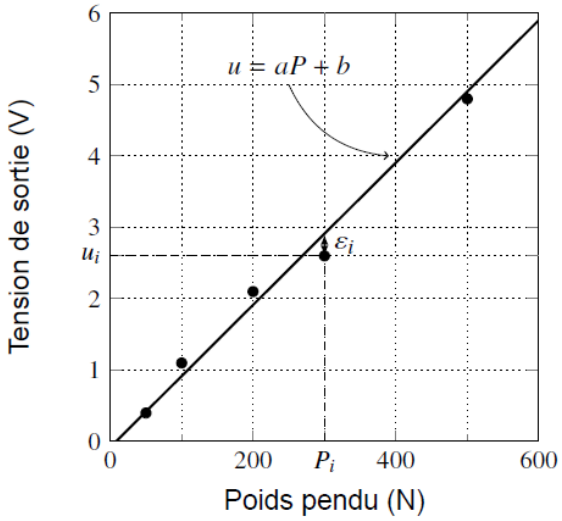



**Figure 9 -** Droite obtenue par la méthode des moindres carrés

L'équation de la droite sera notée $u = aP + b$. Pour chaque mesure, le poids pendu sera noté $P_i$, la tension mesurée $u_i$ et l'écart entre la tension mesurée et la tension modélisée sera noté $\varepsilon_i$ (**figure 9**).

Soit $N$ le nombre de mesures réalisées pendant l'étalonnage ($N = 5$ sur la **figure 9**), la méthode des moindres carrés consiste à trouver les valeurs de $a$ et $b$ qui minimisent le réel :

$$S = \sum_{i=0}^{i=N-1} \varepsilon_i^2.$$

**Q6.** Donner l'expression de $\varepsilon_i$ en fonction de $a$, $b$, $P_i$ et de $u_i$.

La résolution de cette minimisation permet de trouver les expressions de $a$ et $b$ :

$$a = \frac{NS_{Pu} - S_P S_u}{NS_{PP} - S_P^2} \qquad b = \frac{S_P S_{Pu} - S_u S_{PP}}{S_P^2 - NS_{PP}}$$

avec :

$$S_P = \sum_{i=0}^{i=N-1} P_i \qquad S_u = \sum_{i=0}^{i=N-1} u_i \qquad S_{PP} = \sum_{i=0}^{i=N-1} P_i^2 \qquad S_{Pu} = \sum_{i=0}^{i=N-1} P_i u_i.$$

**Notations et hypothèses**

- Les bibliothèques `numpy` et `matplotlib.pyplot` ont été importées : `import numpy as np` et `import matplotlib.pyplot as plt`.
- Les poids $P_i$ sont stockés dans un tableau `numpy` de longueur $N$, noté `P` : `P=np.array([P0,P1,...])`.
- Les tensions $u_i$ sont stockées dans un tableau `numpy` de longueur $N$, noté `u` : `u=np.array([u0,u1,...])`.
- La longueur $N$ n'est pas affectée.

**Fonctions de la bibliothèque `numpy`**

- La fonction `np.dot(X,Y)` donne le produit matriciel ; lorsque `X` et `Y` sont deux tableaux de longueur $N$, `np.dot(X,Y)` a la même valeur que le produit scalaire entre `X` et `Y`.
- La fonction `np.sum(X)` donne la somme des éléments du tableau `X`.
- La fonction `np.ones(N)` donne un tableau `numpy` de longueur $N$ où tous les éléments ont pour valeur 1.

**Q7.** Proposer deux instructions permettant d'affecter les variables `SPP` puis `SPu` représentant $S_{PP}$ et $S_{Pu}$ en utilisant les tableaux `P` et `u`, les fonctions de `numpy` et sans faire de boucle.

**Q8.** Proposer deux instructions permettant d'affecter les variables `SP` puis `Su` représentant $S_P$ et $S_u$ en utilisant les tableaux `P` et `u`, les fonctions de `numpy` et sans faire de boucle.

**Q9.** Proposer des instructions permettant d'affecter les variables `a` puis `b` représentant $a$ et $b$ en utilisant les tableaux `P` et `u` et les variables précédemment définies.

**Q10.** En considérant les variables `a` et `b` affectées, proposer des instructions permettant de tracer uniquement les points et la droite de la **figure 9**. Dans la fonction `plot`, l'argument supplémentaire `'o'` permet de ne pas relier les points par une droite.

### II.2 - Filtrage de la tension de sortie du capteur

***Objectif** : analyser deux types de filtre (exigence 1.3.3.1.2).*

Comme le montre la **figure 10**, la tension de sortie du capteur est bruitée. Afin d'améliorer le comportement de l'Exolift, un filtrage de la sortie du capteur est nécessaire. Deux méthodes de filtrage vont être programmées : la moyenne glissante et le filtre passe-bas.

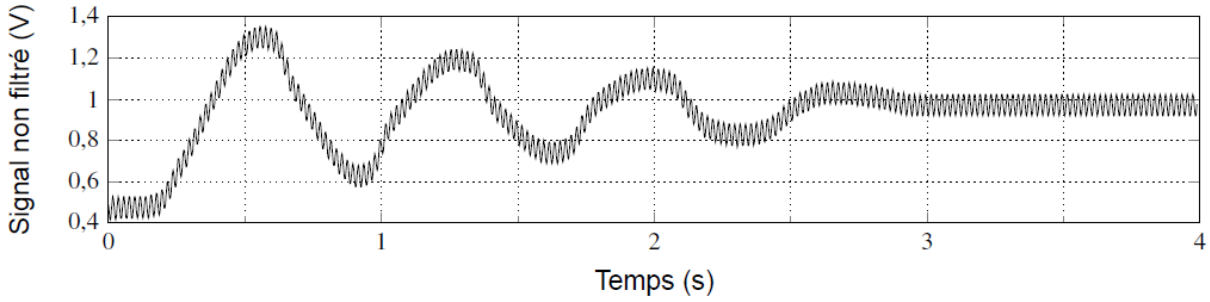



**Figure 10 -** Signal brut du capteur

**Notations et hypothèses**

Les notations et hypothèses de la partie **II.1** sont complétées par :

- les tensions filtrées $u_i^f$ sont stockées dans un tableau `numpy` de longueur $N$, noté `uf` : `uf=np.array([uf0,uf1,...])` ;
- les instants $t_i$ sont stockés dans un tableau `numpy` de longueur $N$, noté `temps` : `temps=np.array([t0,t1,...])`.

### II.2.1 - Filtrage par moyenne glissante

Cette méthode consiste à prendre la moyenne des mesures précédentes. $n$ est le nombre de mesures utilisées pour calculer la moyenne. Il est appelé ordre de la moyenne glissante. La moyenne glissante s'obtient alors par :

$$\text{si } i < n-1,\ u_i^f = \sum_{k=0}^{k=i} \frac{u_k}{i+1} \qquad \text{ou} \qquad \text{si } i \geq n-1,\ u_i^f = \sum_{k=i-n+1}^{k=i} \frac{u_k}{n}.$$

**Q11.** Compléter le tableau du **DR3** en utilisant la méthode de la moyenne glissante pour $n = 3$ avec le tableau `u` donné contenant 5 éléments.

**Q12.** Créer la fonction `filtre_mg(u,n)` qui prend pour argument `u`, un tableau de mesures à filtrer et `n` l'ordre de la moyenne glissante et renvoie un tableau de mesures filtrées par cette méthode.

La **figure 11** présente le résultat du filtrage par la moyenne glissante pour différentes valeurs de $n$.

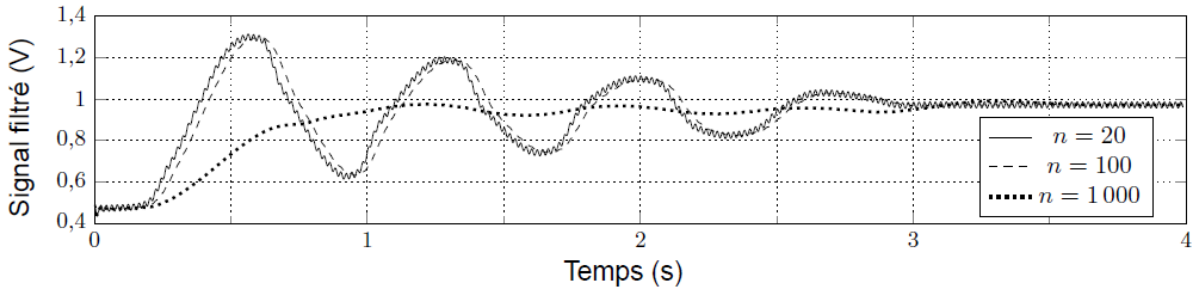



**Figure 11 -** Signal filtré par la méthode de la moyenne glissante pour différents ordres

**Q13.** Quelle est l'influence de l'ordre $n$ sur la qualité du filtrage ?

### II.2.2 - Filtrage par filtre passe-bas

Dans le cas d'un filtre passe-bas d'ordre 1, cette méthode revient à résoudre l'équation différentielle suivante :

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

où $f = \dfrac{1}{2\pi\tau}$ est la fréquence de coupure, $u(t)$ est le signal à filtrer, $u_f(t)$ le signal filtré et $t$ le temps.

**Q14.** Donner l'expression de $u_{i+1}^f$ en fonction de $u_i^f$, $u_i$, $t_{i+1}$, $t_i$ et de $\tau$ en utilisant la méthode d'Euler explicite. Proposer une valeur pour $u_0^f$.

**Q15.** Créer la fonction `filtre_pb(u,temps,f)` qui prend pour argument un tableau `u` des mesures à filtrer, un tableau `temps`, de même dimension, représentant le temps et `f` la fréquence de coupure du filtre passe-bas et qui renvoie un tableau des mesures filtrées par cette méthode.

La **figure 12** présente le résultat du filtrage passe-bas pour différentes valeurs de $f$.

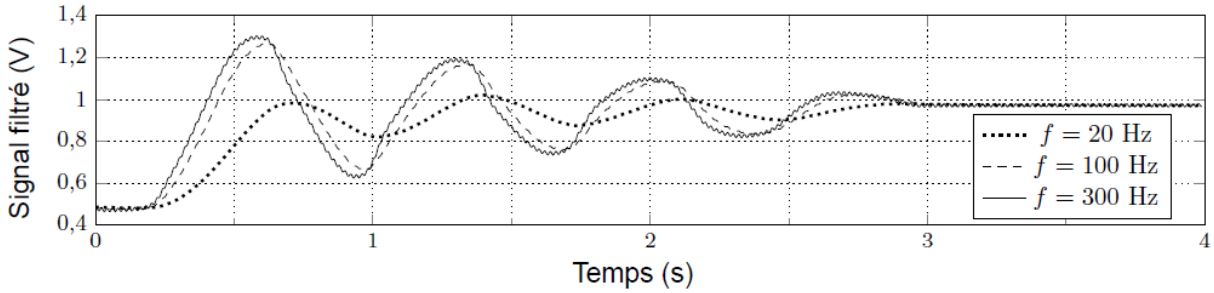



**Figure 12 -** Signal filtré par un filtre passe-bas pour différentes fréquences de coupure

**Q16.** Quelle est l'influence de la fréquence de coupure $f$ sur la qualité du filtrage ?

### II.2.3 - Comparaison des méthodes

La **figure 13** montre le résultat du filtrage par les deux méthodes vues précédemment.

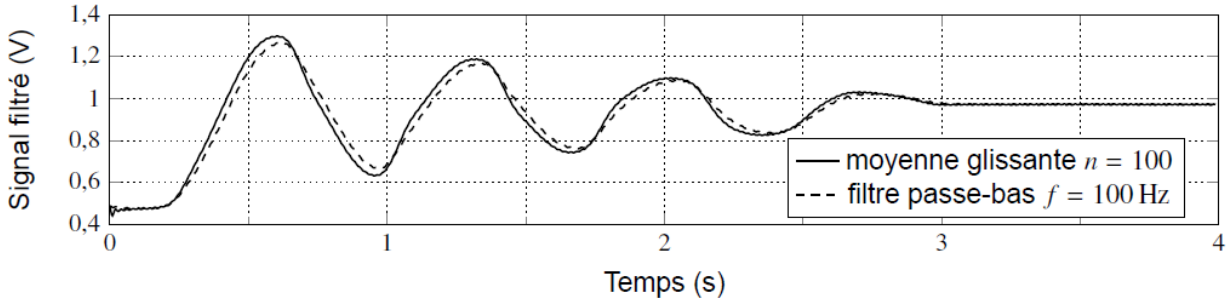


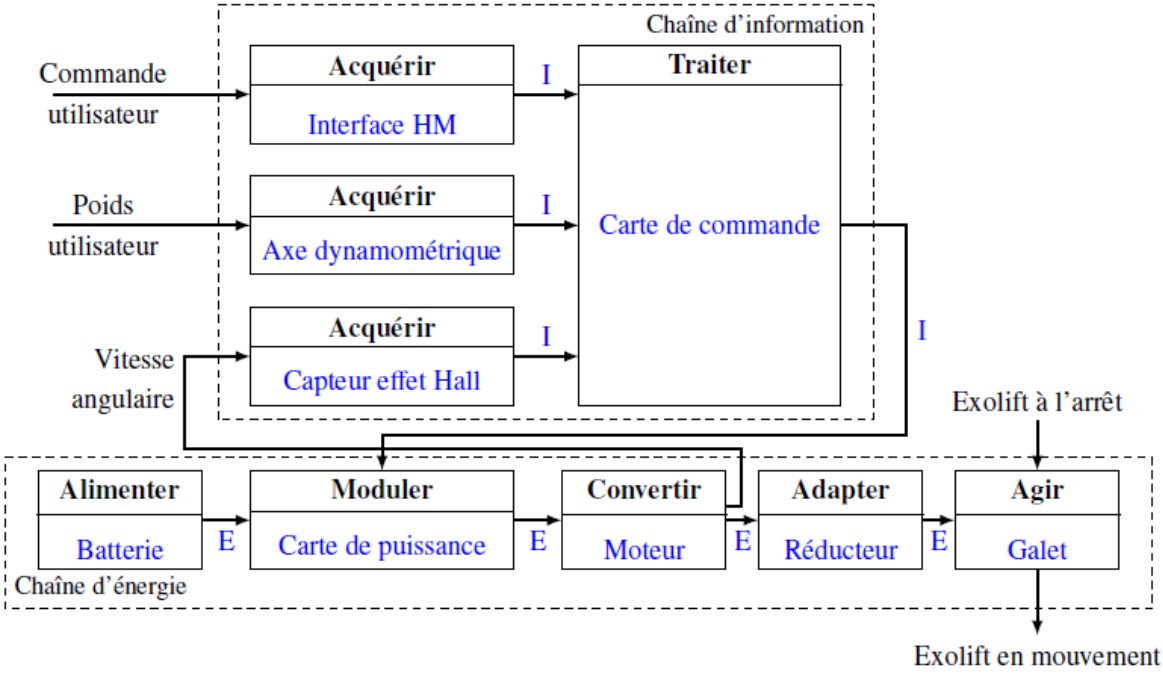

**Figure 13 -** Signal filtré par les deux méthodes

**Q17.** Comparer la complexité temporelle asymptotique des deux méthodes.

CORRIGE

# Partie I - Étude du fonctionnement général du système

**Q1.** **8 pts** : *0,5 pt par composant/flux arrondi au point supérieur (0 si 1 seul)*
Compléter la chaîne structurelle de l'Exolift du **Document Réponse 1** en précisant le nom des composants et la nature des flux échangés (I pour information, E pour énergie et M pour matière).



$$S_{PP} = \sum_{i=0}^{i=N-1} P_i^2$$
$$S_{Pu} = \sum_{i=0}^{i=N-1} P_i u_i$$

```
SPP=np.dot(P,P)
SPu=np.dot(P,u)
```

ou

```
SPP=np.sum(P*P)
SPu=np.sum(P*u)
```

**Q8.** **2 pts** : *1 pt par expression*

Proposer deux instructions permettant d'affecter les variables SP puis Su représentant $S_P$ et $S_u$ en utilisant les tableaux P et u, les fonctions de numpy et sans faire de boucle.

$$S_P = \sum_{i=0}^{i=N-1} P_i$$
$$S_u = \sum_{i=0}^{i=N-1} u_i$$

```
SP=np.sum(P)
Su=np.sum(u)
```

ou

```
SP=np.dot(P,np.ones(len(P)))
Su=np.dot(u,np.ones(len(u)))
```

**Q9.** **3 pts** : *1 pt par expression et 1 pt pour len(P)*

Proposer des instructions permettant d'affecter les variables a puis b représentant $a$ et $b$ en utilisant les tableaux P et u et les variables précédemment définies.

$$a = \frac{NS_{Pu} - S_P S_u}{NS_{PP} - S_P^2}$$
$$b = \frac{S_P S_{Pu} - S_u S_{PP}}{S_P^2 - NS_{PP}}$$

```
N=len(P)
a=(N*SPu-SP*Su)/(N*SPP-SP*SP)
b=(SP*SPu-Su*SPP)/(SP*SP-N*SPP)
```

**Q10.** **3 pts** : *1 pt par instruction, 1 pt pour argument supplémentaire*

En considérant les variables a et b affectées, proposer des instructions permettant de tracer uniquement les points et la droite de la **figure 9**. Dans la fonction `plot`, l'argument supplémentaire `'o'` permet de ne pas relier les points par une droite.

```
plt.plot(P,u,'o')
plt.plot(P,a*P+b)
```

**Q11.** **5 pts** : *1 pt par valeur*

Compléter le tableau du **DR3** en utilisant la méthode de la moyenne glissante pour $n = 3$ avec le tableau u donné contenant 5 éléments.

| $i$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $u_i$ | -2 | 0 | -1 | 2,5 | 3 |
| $u_i^f$ | -2 | -1 | -1 | 0,5 | 1,5 |

**Q12.** **8 pts** : *Voir notation dans le corrigé*

Créer la fonction `filtre_mg(u,n)` qui prend pour argument u, un tableau de mesures à filtrer et n l'ordre de la moyenne glissante et renvoie un tableau de mesures filtrées par cette méthode.

```
def filtre_mg(u,n):
    N=len(u)
    res=np.ones(N)                  # 1 pt
    somme=0                         # 1 pt
    for i in range(N):              # 1 pt
        if i <= n-1:                # 1 pt
            somme+=u[i]
            res[i]=somme/(i+1)      # 1 pt
```

| ui | uif |
|---|---|
| U0 | U0/1=-2 |
| U1 | (U0+U1) /2=-1 |
| U2 | (U0+U1+U2) /3=-1 |
| U3 | (U1+U2+U3) /3=0,5 |
| U4 | ... |

```
      else:
         somme+=u[i]-u[i-n]    # 2 pts
         res[i]=somme/n        # 1 pt
   return res
```

ou

```
def filtre_mg(u,n):
   N=len(u)
   res=np.ones(N)                    # 1 pt
   for i in range(N):                # 1 pt
      somme=0                        # 1 pt
      if i < n-1:                    # 1 pt
         for j in range(i+1)         # 1 pt
            somme+=u[j]
         res[i]=somme/(i+1)          # 1 pt
      else:
         for j in range(i-n+1,i+1)   # 1 pt
            somme+=u[j]
         res[i]=somme/n              # 1 pt
   return res
```

**Q13.** **2 pts** : *1 pt pour petite valeur de n, 1 pt pour grande valeur de n*

Quelle est l'influence de l'ordre $n$ sur la qualité du filtrage ?

Une valeur de $n$ trop petite ne permet pas de filtrer le bruit. Une valeur de $n$ trop grande filtre le bruit mais aussi les variations du signal mesuré : la qualité du filtre est dégradée.

**Q14.** **3 pts** : *1 pt pour* $\frac{du_f(t)}{dt}$, *1 pt pour expression de* $u_{i+1}^f$, *1 pt pour* $u_0^f$

Donner l'expression de $u_{i+1}^f$ en fonction de $u_i^f$, $u_i$, $t_{i+1}$, $t_i$ et de $\tau$ en utilisant la méthode d'Euler explicite. Proposer une valeur pour $u_0^f$.

$$\tau\frac{du_f(t)}{dt} + u_f(t) = u(t) \Rightarrow \tau\frac{u_{i+1}^f - u_i^f}{t_{i+1} - t_i} + u_i^f = u_i$$

$$u_{i+1}^f = u_i^f + \frac{t_{i+1} - t_i}{\tau}\left(u_i - u_i^f\right)$$

On peut choisir $u_0^f = u_0$.

**Q15.** **5 pts** : *Voir notation dans le corrigé*

Créer la fonction `filtre_pb(u,temps,f)` qui prend pour argument un tableau `u` des mesures à filtrer, un tableau `temps`, de même dimension, représentant le temps et `f` la fréquence de coupure du filtre passe-bas et qui renvoie un tableau des mesures filtrées par cette méthode.

```
def filtre_pb(u,temps,f):
   N=len(u)
   tau=1/(2*np.pi*f)                                          # 1 pt
   res=np.ones(N)                                             # 1 pt
   res[0]=u[0]                                                # 1 pt
   for i in range(N-1):                                       # 1 pt
      res[i+1]=res[i]+(temps[i+1]-temps[i])*(u[i]-res[i])/tau # 1 pt
   return res
```

**Q16.** **2 pts** : *1 pt pour petite valeur de f, 1 pt pour grande valeur de f*

Quelle est l'influence de la fréquence de coupure $f$ sur la qualité du filtrage ?

Plus la fréquence de coupure est faible, plus le signal sera filtré entrainant l'effacement des variations du signal en plus du bruit de mesure.
Pour une fréquence trop élevée, le bruit n'est pas filtré.

**Q17.** **3 pts** : *1pt pour complexité par méthode, 1pt conclusion*
Comparer la complexité temporelle asymptotique des deux méthodes.

Moyenne glissante : complexité O(N) (solution 1) ou O(nxN) (solution 2).
Filtre passe bas : complexité O(N) (une seule boucle).
Les deux méthodes se valent en terme de complexité car l'ordre n est rarement élevé.