FILTRAGE DES SIGNAUX NUMERIQUES

Nous allons voir 2 méthodes à connaître pour filtrer les signaux numériques :

- Lissage d'un signal par une moyenne mobile
- Filtre du premier ordre (passe-bas)

1- Moyenne mobile

La moyenne mobile, ou moyenne glissante, est utilisée pour analyser des séries ordonnées de données, le plus souvent des séries temporelles, en supprimant les fluctuations transitoires de façon à en souligner les tendances à plus long terme. Cette moyenne est dite mobile parce qu'elle est recalculée de façon continue, en utilisant à chaque calcul un sous-ensemble d'éléments dans lequel un nouvel élément remplace le plus ancien ou s'ajoute au sous-ensemble.

1-1 Moyenne mobile arithmétique

C'est une **moyenne** qui au lieu d'être calculée sur l'ensemble des n valeurs d'un échantillonnage, est calculée tour à tour sur chaque sous-ensemble de N valeurs consécutives (N << n).
Le sous-ensemble utilisé pour calculer chaque moyenne, parfois appelé « **fenêtre** », « glisse » sur l'ensemble des données.

formule permettant de calculer une moyenne mobile simple :

$$s_n = \frac{1}{N}.\sum_{k=0}^{N-1} e_{n-k} \text{ ou encore } s_n = s_{n-1} + \frac{e_n - e_{n-N}}{N}$$

1-2 Moyenne mobile pondérée, moyenne mobile exponentielle

On peut définir des moyennes mobiles qui vont attacher un poids plus important aux données les plus récentes.

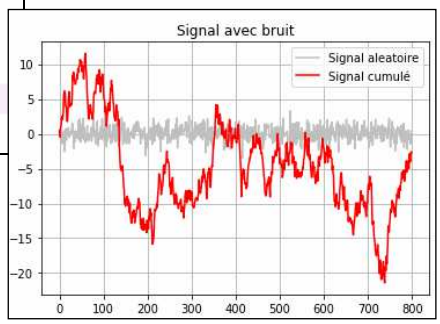
2- Génération du signal pour les tests

On génère un signal aléatoire qui sera ensuite filtré.
Pour retrouver un signal ressemblant à un signal bruité, on fait la somme des données cumulées.

```
import numpy as np
from numpy.random import randn
import matplotlib.pyplot as plt

signal0 = (randn(800))  # Brownian noise
plt.plot(signal0, color='silver', label='Signal aleatoire')
plt.grid(True, which='both')
plt.legend(loc="best")
plt.title("Signal avec bruit")

signal2 = np.cumsum(randn(800))  # Brownian noise
plt.plot(signal2, color='red', label='Signal cumulé')
plt.grid(True, which='both')
plt.legend(loc="best")

plt.show()
```

On génère un signal aléatoire

On fait la somme cumulée des données aléatoires



3- Lissage d'un signal par une moyenne mobile ¶

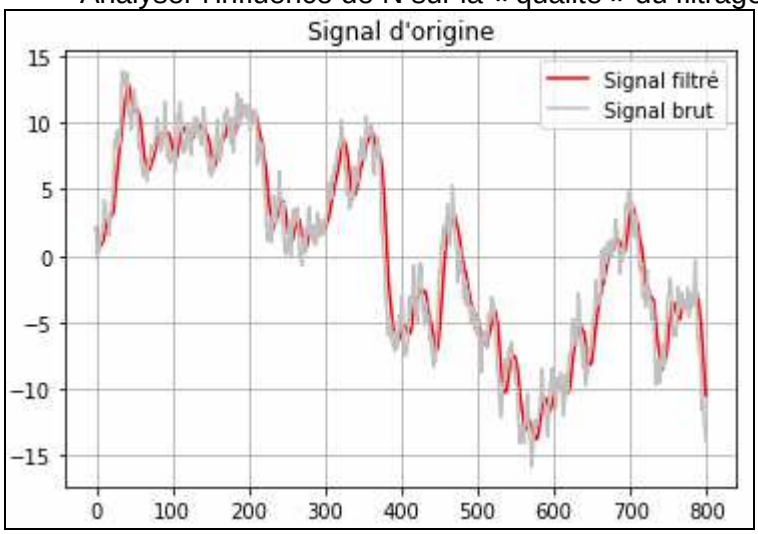
Mettre en œuvre le script python permettant de lisser avec une moyenne mobile **signal2**.

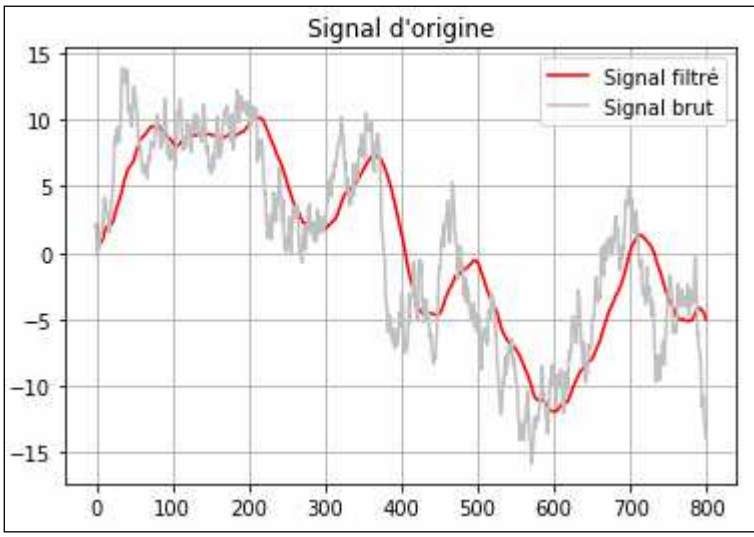
Vous pouvez reprendre les solutions proposées dans le corrigé du DM2 pour

Faire tracer sur le même graphique signal2 et le signal filtré

M Salette- Lycée Brizeux- Quimper

Analyser l'influence de N sur la « qualité » du filtrage





Rappel des données du corrigé du DM2 : (2 versions proposées)

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

```
def filtre_mg(u,n):
    uf=np.ones(len(u))
    for i in range(len(u)):
        if i<n-1:
            uf[i]=(np.sum(u[:i+1])/(i+1))
        else:
            uf[i]=(np.sum(u[i-n+1:i+1])/n)
        return uf
```

Remarque : il n'est pas nécessaire d'utiliser une fonction ici

4- Filtre passe-bas¶

Filtre passe-bas du 1er ordre¶ : La fonction de transfert d'un filtre passe-bas du premier ordre est : $H(p)=\frac{1}{1+\tau.p}$ avec $\frac{1}{\tau}$ la pulsation de coupure du filtre.

L'équation différentielle associée à ce filtre est : $\tau.\frac{ds(t)}{dt}+s(t)=e(t)$

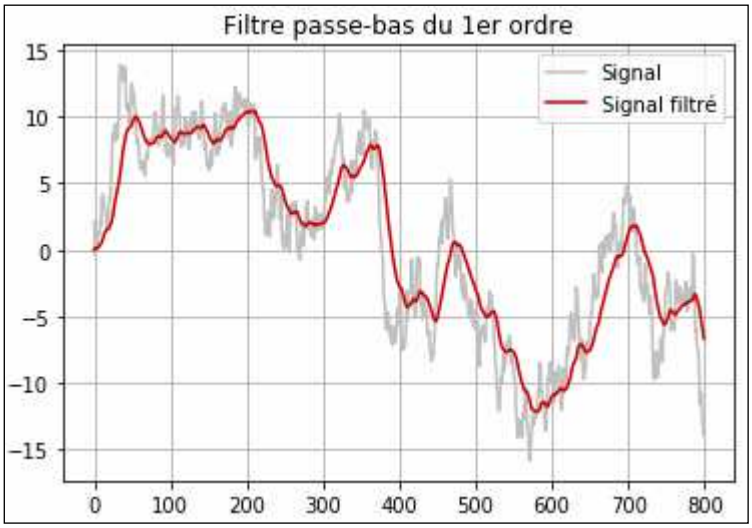
L'équation discrétisée devient alors : $\tau.\frac{s_{n+1}-s_n}{T_e}+s_n=e_n$

L'équation permettant de calculer s_n quel que soit n devient : $s_n=s_{n-1}+\frac{T_e}{\tau}.(e_{n-1}-s_{n-1})$ avec $s_0=0$

Implémentation sous python

```
# Fréquence de coupure
fc = 0.1  # Hz
tau = 1/(2*np.pi*fc)
# Période d'échantillonnage
Te = 1  # s
# Préparation de la liste de sortie
s_pb = []
s_pb.append(0)
# Application du filtre
for i in range(1, len(signal_bruit)):
    s_pb.append(s_pb[i-1]+Te/tau*(signal_bruit[i-1]-s_pb[i-1]))
```

Analyser l'influence de la fréquence d'échantillonnage et de la pulsation de coupure sur la « qualité » du filtrage



5- Exporter et importer les données

Pour pouvoir comparer les différents filtres, il est préférable de travailler sur le même signal .
Il faut donc pouvoir stocker les données du signal avec bruit. Cela peut se faire sous forme de fichier .csv (tableur) ou .txt (texte brut)

Les commandes qui suivent permettent de créer un fichier txt à partir des données de la liste ***signal***.
Ce fichier est créé à l'emplacement où est enregistré le script

```
#Création du fichier
with open("file.txt", 'w') as f:
    for s in signal:
        f.write(str(s) + '\n')
```

Les commandes qui suivent permettent de recréer une liste à partir de ce fichier .txt,

```
#reouverture du fichier
fichier= open ("file.txt","r")
stockage=fichier.readlines()
fichier.close()
fichier_lu=[]
for i in stockage:
    valeurs=i.split()
    fichier_lu+=[float(valeurs[0])]
```

Lancer la génération de signal avec bruit et retenir une réponse pertinente pour la suite de l'étude.
Reprendre les scripts précédents en faisant appel à cette réponse afin d'analyser l'influence des paramètres sur la qualité du filtrage.