# TD : Application de l'algorithme des $k$-moyennes à la quantification de signaux

## Historique

La méthode des k-moyennes a été développée par Lloyd, Stuart P. en 1957 pour les laboratoires de téléphonie Bell afin d'optimiser la représentation en modulation par impulsions et codage d'un signal analogique. Cette dernière se décompose en trois opérations successives: : échantillonnage, quantification puis codage.

L'algorithme de quatification proposé par Lloyd se base sur le constat que les valeurs des niveaux (quanta) devraient être espacées plus étroitement dans les régions de tension où l'amplitude du signal est plus susceptible de se situer.

## Rappel : Qu'est-ce que la quantification ?

La **quantification** est un processus utilisé en traitement du signal pour représenter des valeurs continues par un ensemble fini de niveaux. Cela est particulièrement utile dans des domaines comme la compression et la transmission des données.

### Exemple simple :

Imaginez que vous devez transmettre des valeurs de température exterieure mesurées avec une précision de $1°C$. Si vous choisissez de les représenter avec seulement trois niveaux (par exemple : $-10°C$, $10°C$, et $30°C$), vous perdez en précision, mais vous simplifiez la représentation et réduisez la taille des données.

### Étapes principales de la quantification :

1. **Découpage en niveaux :** Les valeurs possibles du signal sont réparties en $k$ groupes (ou niveaux). Classiquement, ces niveaux sont équirépartis sur la plage de valeurs possibles.

2. **Affectation :** Chaque valeur du signal est attribuée au niveau le plus proche.

3. **Reconstruction :** À partir des niveaux attribués, on peut reconstruire une version simplifiée du signal.

# 1 Quantification d'un signal

Le signal analogique sur lequel nous travaillerons aujourd'hui est un signal sinusoïdal bruité défini par la relation suivante :

$$s(t) = \sin(2\pi t) + \epsilon$$

Dans un premier temps, nous devons échantillonner ce signal. $t \in [0, 1]$ est échantillonné en 10 points, et $\epsilon$ est un bruit aléatoire gaussien centré en 0.

Vous utiliserez $k = 3$ pour quantifier ce signal.

## 1.1 Génération d'un signal échantillonné

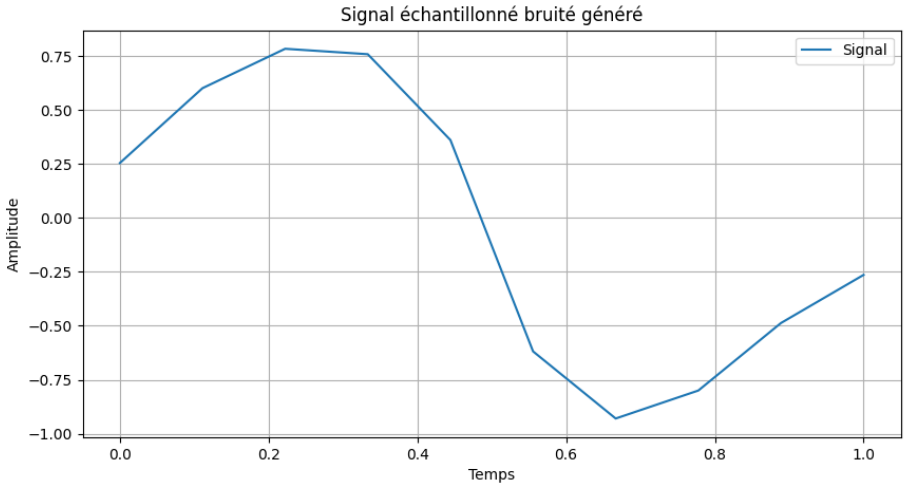
**Proposer une fonction pour générer un signal echantillonné en fonction du nombre de points et du niveau de bruit.**
On pourra utiliser la fonction *numpy.random.normal* pour créer un tableau de points distribués suivant une loi normale. Elle prend comme arguments la moyenne de l'échantillon à générer, son écart type et le nombre de points.
Vous pourrez utiliser la fonction *numpy.linspace* pour générer un tableau de nombres également espacés sur un intervalle. Elle prend comme arguments la borne inférieure incluse, la borne supérieure exclue et le nombre de nombres dans l'array renvoyé.
Pour ce faire, vous pouvez compléter le squelette suivant :

```
def generer_signal_ech(n_points, noise_level):
    """
    n_points : Nombre de points à générer
    noise_level : Ecart type du bruit ajouté
    return: Les valeurs du signal bruité

    """
    # Compléter ici
```

**Que doit-on vérifier pour garantir l'intégrité de l'échantillonnage ?**

Le signal obtenu après échantillonnage est le suivant :



Voici les valeurs de $t$ et de $s_{ech}(t)$ obtenues :

| $t$ | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|---|
| $s_{ech}(t)$ | 0.25 | 0.62 | 0.77 | 0.75 | 0.38 | −0.63 | −0.88 | −0.79 | −0.48 | −0.25 |

## 1.2 Quantification uniforme

La quantification uniforme est la méthode la plus plus simple, celle que vous connaissez déjà : les niveaux de quantification sont répartis uniformément et distants du même pas de quantification.

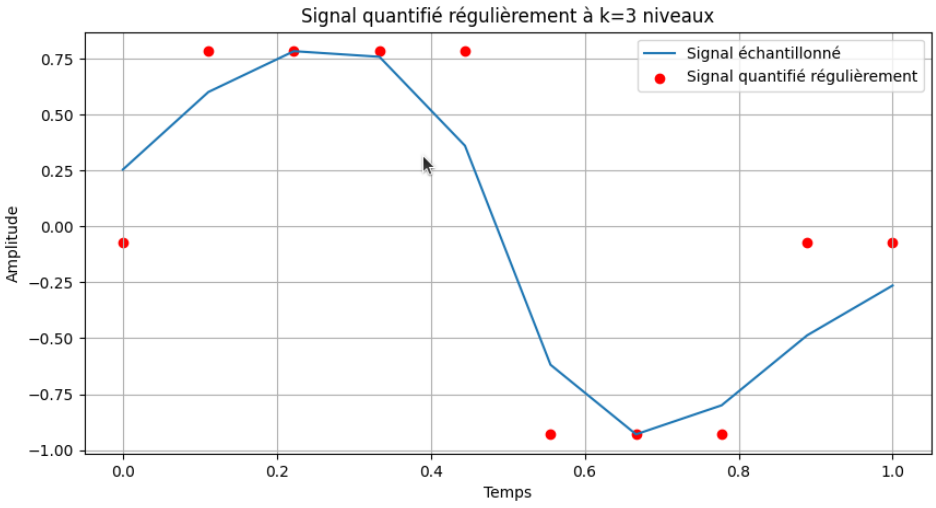
**Proposer une fonction permettant de générer le signal quantifié à partir du signal échantillonné et du nombre de niveaux de quantification.**
Completer le squelette suivant :

```
def quantification_regular(signal_bruite, k):

 """
 signal_bruite : Signal échantillonné à quantifier
 k : Nombre de quanta
 return: Les valeurs du signal quantifié

 """
 # Compléter ici
```

Le signal obtenu est le suivant :




## 1.3 Quantification et $k$-moyennes

L'algorithme des $k$-moyennes est bien adapté pour résoudre le problème de quantification :

- Les $k$ centres des clusters représentent les niveaux de quantification.

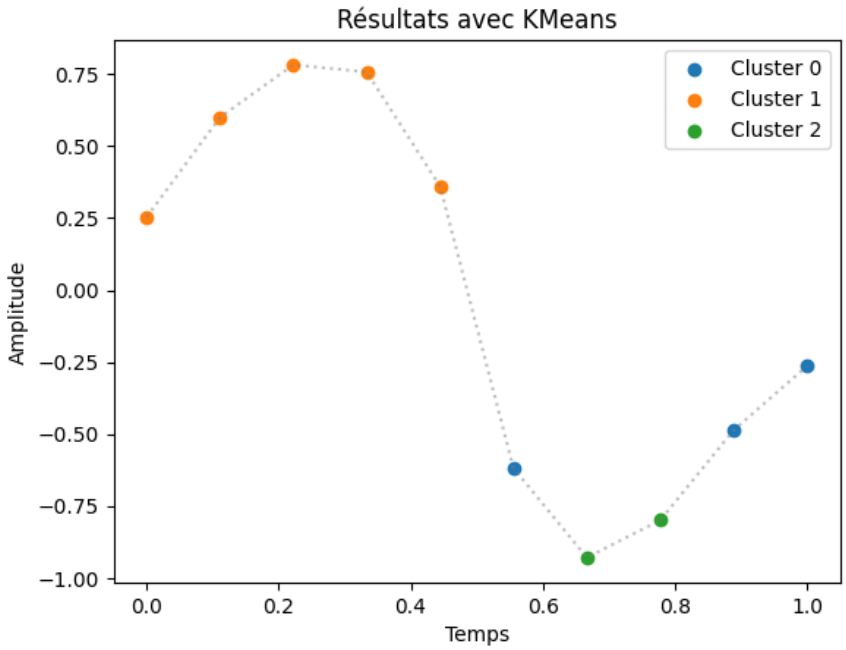
- L'affectation des points au centre le plus proche correspond à l'étape de quantification proprement dite.
- La mise à jour des centres permet d'optimiser les niveaux de manière à minimiser l'erreur globale.

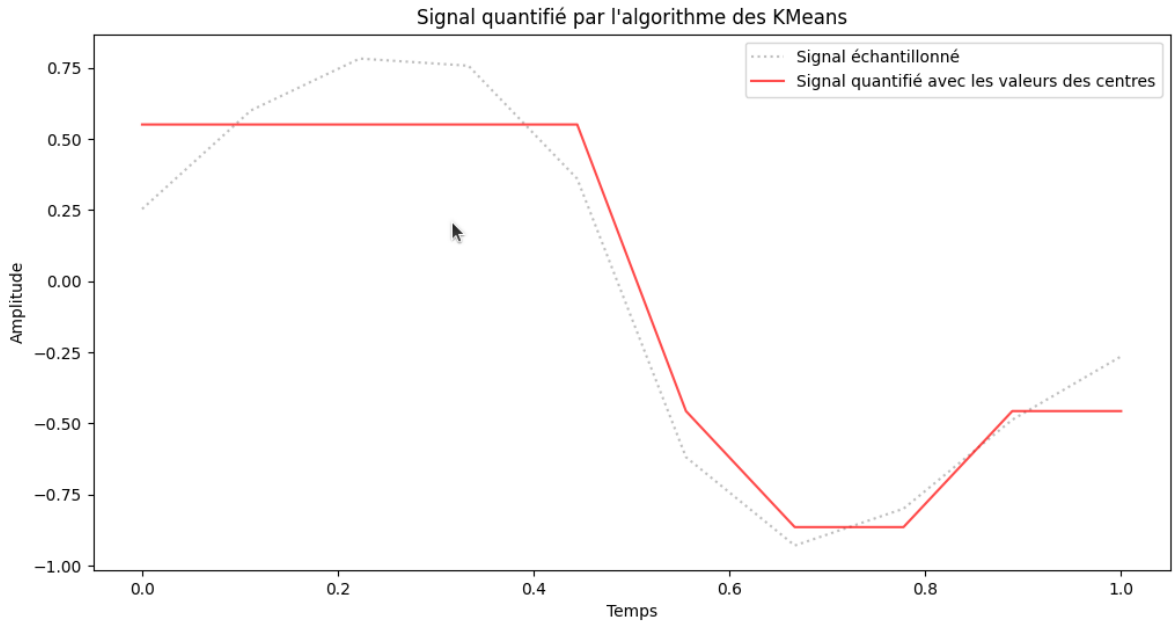
Le problème de calcul peut être formalisé comme suit:

- **Entrée :** Une suite de valeurs flottantes dans une plage $P$.
- **Sortie :** Un ensemble $S$ d'élements de $P$, de cardinal $k$ et une fonction $f : P \to S$

A partir de l'implémentation de l'algorithme des k-moyennes vue en cours, **proposer un programme permettant de retourner la valeur finale des centres ainsi que le signal quantifié avec ces derniers.** Vous disposez de toutes les fonctions du cours et du signal échantillonné.

Le signal obtenu est le suivant :






```
Signal echantillonné :  [ 0.25383385  0.60039297  0.78277376  0.75758965  0.36026001 -0.61829509
 -0.92909368 -0.79954585 -0.48693121 -0.26413141]
Centres calculés après KMeans :  [[-0.45645257083341323], [0.5509700473993663], [-0.8643197649376146]]
Signal remplacé par les centres :  [ 0.55097005  0.55097005  0.55097005  0.55097005  0.55097005 -0.45645257
 -0.86431976 -0.86431976 -0.45645257 -0.45645257]
```

### 1.4 Reconstruction du signal

Après avoir appliqué l'algorithme des $k$-moyennes, vous avez obtenu :

- les centres des clusters : $[-0.46, 0.55, -0.86]$ ;
- les labels attribués à chaque point du signal : $[1, 1, 1, 1, 1, 0, 2, 2, 0, 0]$.

**Compléter la fonction suivante pour reconstruire le signal à partir des résultats des $k$-moyennes** :

```
def reconstruct_signal(labels, centers):
"""
Reconstruit le signal à partir des clusters et des centres.
labels : liste des indices de centres pour chaque point.
centers : liste des valeurs des centres.
"""
# Compléter ici
```

## 2 Comparaison des deux méthodes de quantification

### 2.1 Calcul de l'erreur quadratique moyenne (MSE)

L'erreur quadratique moyenne est une métrique permettant, dans ce contexte, de représenter la quantité d'information perdue au cours de l'échantillonnage, elle est définie par :

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(s_i - \hat{s}_i)^2$$

où $s_i$ est la valeur d'origine et $\hat{s}_i$ la valeur reconstruite.
**Proposer une fonction pour calculer la moyenne des erreurs quadratiques entre le signal original et le signal reconstruit.** Compléter le squelette suivant :

```
def compute_mse(original, reconstructed):
"""
original : signal d'origine.
reconstructed : signal reconstruit.
"""
# Compléter ici
```

**Commenter les résultats obtenus ci-dessous** :

```
MSE avec quantification régulière: 0.0640394960053124
MSE avec KMeans : 0.029602818704592294
```

## 2.2 Limites de l'algorithme

1. Quel est l'impact du choix de $k$ sur la qualité de la quantification ?
2. En pratique, quelles valeurs de k va-t-on employer ? Justifiez votre réponse
3. Dans quels cas l'algorithme des $k$-moyennes peut-il échouer à trouver une bonne quantification ?
4. Quelles alternatives pourriez-vous proposer pour surmonter ces limites ?