Année universitaire
2025 - 2026

S7 — Traitement des Signaux Aléatoires
Projet

Déroulement. Ce projet est à réaliser en binôme, durant 6 séances de 4 heures.

Configuration. Le TP est à réaliser en python, et à exécuter dans l'environnement virtuel `env_msi`. Un template du projet à compléter est fourni sur e-Campus.

Pour mettre à jour la librairie `msicpe` :

- Ouvrir un terminal ou une invite de commandes et se placer dans le répertoire où l'environnement `msicpe` a été installé :
  `cd chemin_dossier_contenant_env_msi`
- Activer l'environnement virtuel :
  Pour Windows : `.\env_msi\Scripts\activate`
  Pour Linux/Mac : `source env_msi/bin/activate`
- Exécuter la commande suivante dans le même terminal :
  `python -m pip install -U msicpe`

Ressources. Vous avez à votre disposition la documentation technique de msicpe, numpy, plotly express, scipy, pandas, vos cours/notes, ...

Évaluation. L'évaluation du projet portera sur :

- une note de restitution orale finale de 20 minutes, au format **imposé par le template** :
  - 1 slide de contexte,
  - 1 slide de synthèse (carte mentale),
  - 1 ou 2 slides de présentation de chaque étape de la chaîne de traitement.
- une note de QCM

L'ensemble de ces 2 notes constituera la note de TP.

> Ce projet a pour vocation d'être formateur, et peut nécessiter de travailler en groupe, ou d'utiliser des ressources externes ; en particulier, des documents trouvés sur internet ou des outils d'intelligence artificielle.
> Le recours à ces ressources n'est pas interdit dans le cadre de ce TP.

Toutefois, cela ne vous affranchit pas de faire preuve d'analyse et d'esprit critique. Vous devez également être en mesure d'expliquer en détail chaque ligne du code que vous aurez implémenté. À ce titre, l'évaluation portera sur **votre** compréhension, restitution des notions et éléments du code. Tout travail emprunté à un (ou plusieurs) autre(s) groupe(s), trouvé sur internet ou dans un livre, ou encore généré par intelligence artificielle, doit être **explicitement identifié** et **correctement référencé**.

Tout travail ne respectant pas ces conditions sera considéré comme non recevable.

Feuille de route du projet

Cahier des charges

Contexte du projet

On dispose d'un signal à transmettre, selon un contexte et des modalités à choisir parmi les sujets suivants :

- **Sujet 1 : Déterminer des lieux d'implantation de captation d'énergies renouvelables.** Pour maximiser la production d'énergies renouvelables, le syndicat des énergies renouvelables (SER) souhaite analyser différentes zones en fonction des ressources naturelles disponibles (ensoleillement, vent, eau, ...). On s'intéressera dans ce projet à la transmission via liaison satellite d'enregistrements provenant de différents capteurs (vitesse du vent, rayonnement solaire) enregistrés en 2016 dans la région Auvergne-Rhône-Alpes (voir fichier `AURA_rayonnement-solaire-vitesse-vent-tri-horaires-regionaux.csv`).

- **Sujet 2 : Déterminer la faisabilité de la télésurveillance par EEG.** Les technologies de surveillance médicale à distance permettent de suivre en continu l'état de santé des patients en dehors des hôpitaux. Dans ce cadre, l'électroencéphalographie (EEG) s'avère particulièrement utile dans le suivi de maladies neurologiques telles que l'épilepsie ou les troubles du sommeil. Cependant, l'utilisation de l'EEG en télésurveillance nécessite une bonne qualité de transmission, afin de garantir la qualité des signaux reçus à distance. En l'occurrence, le signal disponible dans le fichier `EEG.csv` sera transmis via WiFi.

- **Sujet 3 : Améliorer la qualité audio des captations de concerts.** La captation en haute qualité est un défi en raison notamment des interférences sonores et des limites techniques des équipements de captation. On s'intéresse dans ce projet à une mélodie (`ProtestMonoBruitTronque.wav`) captée par un microphone, et transmise via un câble à l'enregistreur.

- **Sujet 4 : Preuve de concept d'un système de maintenance autonome pour des espaces verts.** Les paysagistes Sauvaje souhaitent étudier la pertinence de développer un robot jardinier capable de prendre soin de plantes de manière autonome, grâce à une communication continue par Bluetooth avec un réseau de capteurs environnementaux (température, humidité, déficit de pression vapeur, ...). La qualité de la transmission est donc primordiale dans ce contexte. Les relevés de ces différents capteurs sont disponibles dans le fichier `Sensor_data.csv`.

Durant ce projet, il s'agira donc de comprendre, mettre en œuvre, et analyser la chaîne de traitement complète (incluant la transmission du signal choisi) proposée à la figure 1. Nous considérerons dans ce projet que **la transmission du signal est perturbée par un bruit** qui altère l'intégrité de l'information transmise. **La présence de ce bruit justifie l'utilisation des méthodes vues en cours de traitement de signal aléatoire afin de restituer l'information originale.**

Toutes les notations utiles au projet sont regroupées dans le tableau de synthèse 1 ci-dessous, qu'il conviendra de compléter au fur et à mesure du projet.

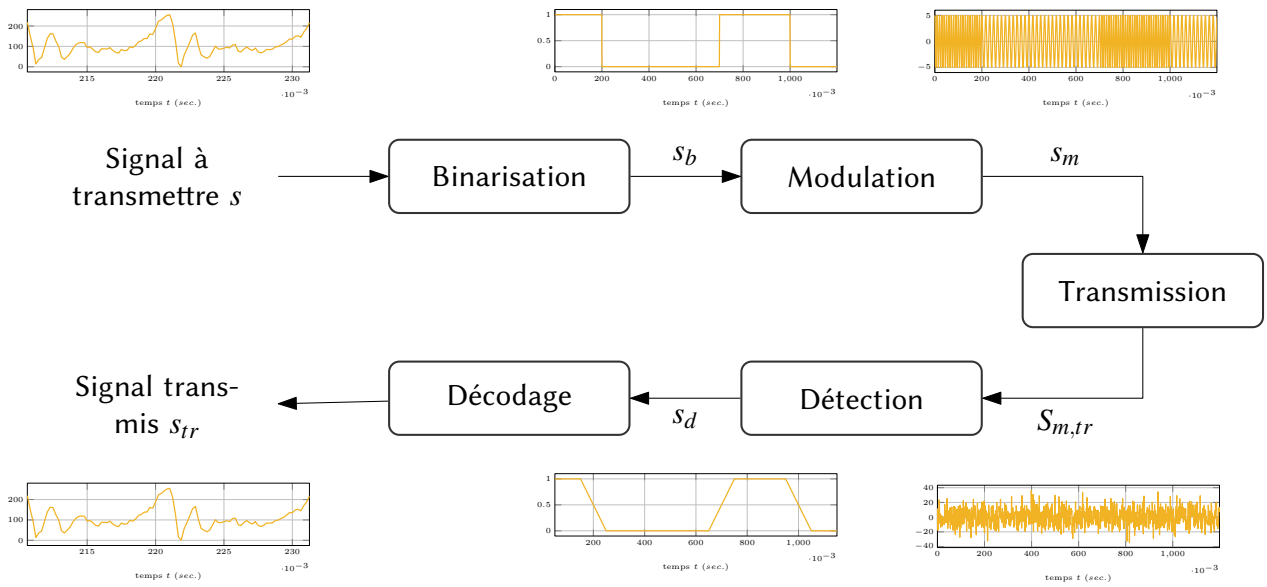


FIGURE 1 – Schéma global du projet

Signal	**Binaire ?**	**Variable**	**Nb points**	**Fréquence d'échantillonnage**	**Vecteur temps**
s					
s_b					
s_m					
$s_{m,tr}$					
s_d					
s_{tr}					

TABLE 1 – Tableau récapitulatif des différents signaux de la chaîne de traitement

La feuille de route du projet est la suivante :

Étape 1. Il faut dans un premier temps **moduler le signal** (binarisé), pour l'adapter au canal de transmission. On utilisera la Modulation par Déplacement de Fréquence Binaire (MDFB), qui consiste à associer une fréquence prédéterminée aux différents symboles du message d'entrée ('0' ou '1' dans notre cas).

Le signal ainsi transformé est ensuite transmis par un canal de transmission, qui introduit généralement du bruit. Parfois, la dégradation est tellement importante (introduite par le canal lui-même, d'éventuelles interférences extérieures...) que le signal utile se retrouve noyé dans le bruit.

Étape 2. Le bruit introduit par le canal de transmission est de nature inconnue. Afin d'adapter les traitements au type de bruit, il est donc nécessaire de le **caractériser** statistiquement, dans un premier temps dans le **domaine direct** (temporel ou spatial). L'estimation des moments (moyenne, puissance, ...) et de la densité (gaussienne, poisson, ...) se fera via l'histogramme.

Étape 3. Il est aussi nécessaire de **caractériser spectralement** le bruit. Cela est aussi utile pour définir un profil de filtre (cf dernier TP de TNS en 3ETI).

Étape 4. Après avoir caractérisé le bruit, on peut donc procéder à la **détection du signal utile** dans le bruit. Pour cela, on mettra en place un filtrage adapté, pour chercher dans le signal transmis (bruité) s'il y a présence ou non du signal utile envoyé (qui est connu). Une fois que le signal utile a été détecté il sera, dans notre cas, nécessaire de reconstruire le signal binarisé. La dernière étape consiste à **décoder** le signal, c'est-à-dire à transformer le signal binaire en sa représentation décimale.

Étape 5. La détection du signal transmis a pu introduire des erreurs dans le signal décodé (par exemple, une mauvaise détection entraînant une confusion entre symboles '0' et '1'). On se propose donc de **restaurer le signal initial** en débruitant le signal reconstruit. Pour cela, on comparera différentes possibilités pour réaliser cette tâche (filtre passe-bas, modèle auto-régressif d'ordre M).

Objectifs du projet

- Concevoir la simulation d'un problème réel complexe
- Estimer les grandeurs caractéristiques d'un signal aléatoire
- Mettre en œuvre un filtre linéaire pour différents objectifs (détection, restauration, ...)
- Résumer les notions incluses et/ou mises en œuvre en utilisant des outils visuels
- Restituer succinctement certains résultats marquants

Modulation par Déplacement de Fréquence Binaire (MDFB)

Objectifs pédagogiques de la séance

- Expliquer le principe de la modulation MDF(B)
- Mettre en œuvre la binarisation puis la MDF(B) d'un signal aléatoire
- Examiner les signaux modulés pour valider l'implémentation

Dans un premier temps, il faut binariser puis moduler le signal, pour l'adapter au canal de transmission.

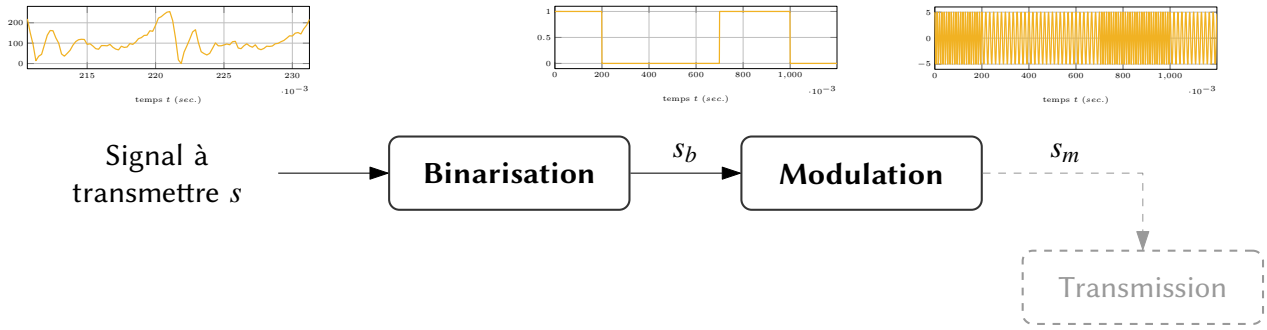


FIGURE 2 – Étape de modulation

Étude théorique de la MDFB

Binarisation

En informatique, les types de données (chaînes de caractère, nombres entiers, décimaux, ...) sont représentés différemment (voir Table 2). À toutes fins utiles, on rappelle qu'un octet est un ensemble de 8 bits.

Type	Intervalle	Taille (octets)
char	-	1
int8	-128,127	1
uint8	0,255	1
float32	$\pm$ 3.4028235 $\times$ 10^{38}	4
double	$\pm$ 1.7976931348623157$\times$ 10^{308}	8

TABLE 2 – Données numériques (type, intervalle, taille binaire en octets)

Aussi, dans le cadre de ce projet, il est nécessaire de binariser (i.e. convertir en représentation binaire) les données pour les transmettre via le canal de transmission. Ainsi, l'étape initiale du projet sera de déterminer quel est le type des données afin de les binariser correctement. De plus, dans notre cas, le débit binaire d_{bin} (en bit/seconde) du canal de transmission sera fixé à 10 bits/seconde.

Q 1.1 : Pour chaque type de donnée numérique, déterminer le débit de données (en type/sec) en fonction du débit binaire d_{bin}.

Modulation par Déplacement de Fréquence (MDF)

La modulation par déplacement de fréquence (MDF) est un mode de modulation de fréquence numérique dans lequel la fréquence du signal modulé varie entre des fréquences prédéterminées. Dans notre cas d'étude, nous utilisons donc deux fréquences particulières pour transmettre une information binaire (des '1' et des '0').

Un bit de valeur '1' est encodé par une sinusoïde de fréquence ν_1 sur une durée T_{bin} contenant N_{mod} points échantillonnés à la fréquence $F_{mod} = 1/T_{mod}$ (voir Figure 3). Autrement dit, un bit de valeur '1' sera remplacé par le signal :

$$s_1[k] = A\cos\left(2\pi\nu_1 k T_{mod}\right), \quad k \in [\![0, 1, \ldots, N_{mod} - 1]\!], \tag{1}$$

avec A l'amplitude du signal.
De même, un bit de valeur '0' sera encodé par une sinusoïde de fréquence ν_0 sur une durée T_{bin} contenant N_{mod} points. Autrement dit, remplacé par :

$$s_0[k] = A\cos\left(2\pi\nu_0 k T_{mod}\right), \quad k \in [\![0, 1, \ldots, N_{mod} - 1]\!]. \tag{2}$$

En pratique, on devrait idéalement choisir la période d'échantillonnage du signal binaire T_{bin} (et donc la durée d'un train d'onde s_0 ou s_1) telle que s_0 et s_1 contiennent un nombre entier ($\gg 1$) de périodes.

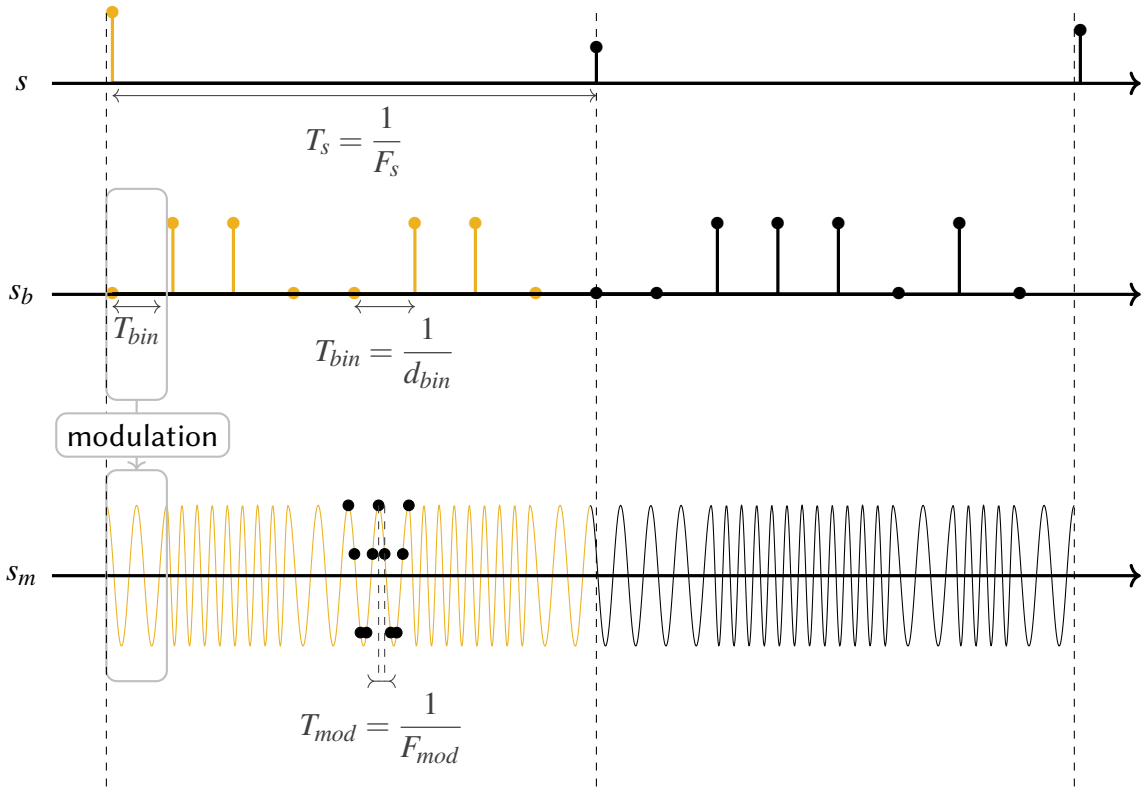


FIGURE 3 – Principe de la modulation par déplacement de fréquence binaire (MDFB).

Q 1.2 : Représenter sur un même schéma les DSE des signaux modulés correspondant à chacun des 2 symboles '0' et '1'.

Q 1.3 : Quelles relations les fréquences F_{mod}, ν_0, ν_1 et la durée T_{bin} des trains d'onde doivent-elles vérifier ?

Q 1.4 : À ν_0 et ν_1 fixées, quel est alors le taux de transmission $d_{bin} = 1/T_{bin}$ (débit binaire) maximal envisageable ?

Q 1.5 : Exprimer finalement le nombre d'échantillons N_{mod} dans un train d'onde de durée T_{bin} échantillonné à la fréquence F_{mod} ?

Implémentation de la MDFB

Typologie de Données

Q 1.6 : Selon le sujet choisi, identifier le type de données du signal ainsi que sa fréquence d'échantillonnage F_s.

Charger les données, à l'aide la fonction `msicpe.tsa.load_signal`.

Afin d'éviter des lenteurs lors de l'exécution du code, on travaillera seulement sur maximum les $N = 1500$ premiers points du signal choisi. Restreindre le signal chargé s à ses N premiers points.

On rappelle que la documentation peut être trouvée en ligne (voir msicpe).

Binarisation

Créer le signal s_b par binarisation des échantillons de s. Pour cela, on utilisera la fonction `msicpe.tsa.data2bin`. La variable t_b contiendra quant à elle les instants liés à l'échantillonnage du signal binaire s_b, créée à partir du débit binaire d_{bin} fixé à 10 bits/sec.

Remarque : Nous ne considérons pas le cas de la transmission en temps réel dans le cadre du projet, sans quoi il faudrait choisir un débit binaire d_{bin} qui dépend de la fréquence d'échantillonnage F_s de chacun des signaux.

Modulation

On choisira comme paramètres de modulation :

- la fréquence d'échantillonnage du signal modulé : $F_{mod} = 1$ kHz,
- l'amplitude du signal modulé A : 5V,
- la fréquence ν_0 définie pour (2) : $\nu_0 = 20$ Hz,
- la fréquence ν_1 définie pour (1) : $\nu_1 = 40$ Hz.

En utilisant les équations (1) et (2), générer le signal modulé s_m à partir de :

- s_b : le signal binaire,
- F_{mod} : la fréquence d'échantillonnage du signal modulé,
- ν_0 : la fréquence du signal sinusoïdal encodant le symbole '0',

- ν_1 : la fréquence du signal sinusoïdal encodant le symbole '1',
- d_{bin} : le débit binaire.

Créer un vecteur temps t_m contenant les instants d'échantillonnage du signal modulé s_m.

Reproduire la figure 3 pour une durée équivalente à 16 bits du signal s_b.

L'affichage d'un signal possédant un très grand nombre de points peut faire crasher VSCode. Aussi, pour chaque signal que vous souhaiterez afficher, pensez à réduire sa dimension **dans la dataframe** créée pour l'affichage :

```
Npts_to_plot = ...
df_s = pd.DataFrame({'x': ts[:Npts_to_plot], 'y':
 ↪  s[:Npts_to_plot], 'legende':'s'})
...

Nbits_to_plot = ...
df_sb = pd.DataFrame({'x': tb[:Nbits_to_plot], 'y':
 ↪  sb[:Nbits_to_plot], 'legende':'sb'})
...
```

Q 1.7 : Combien de bits sont encodés dans 1 seconde du signal s_m ?

Estimation de la Densité De Probabilité (DDP)

Objectifs pédagogiques de la séance

- Connaître un estimateur de la DDP d'un signal aléatoire
- Mettre en œuvre numériquement un estimateur de la DDP
- Examiner l'influence des hyperparamètres de l'estimateur sur sa qualité (biais, variance)
- Déterminer la DDP d'un signal aléatoire et ses caractéristiques statistiques associées

Une fois le signal modulé, il est transmis jusqu'au récepteur. Cette transmission n'est pas neutre vis-à-vis du signal, qui peut subir des atténuations (qu'on ne considérera pas ici) et être parasité par du bruit. Il est donc important de caractériser (i.e. déterminer les propriétés statistiques et fréquentielles de) la dégradation ajoutée par le canal de transmission.
Nous allons dans un premier temps nous concentrer sur la caractérisation statistique (DDP, moments, ...) du canal de transmission.

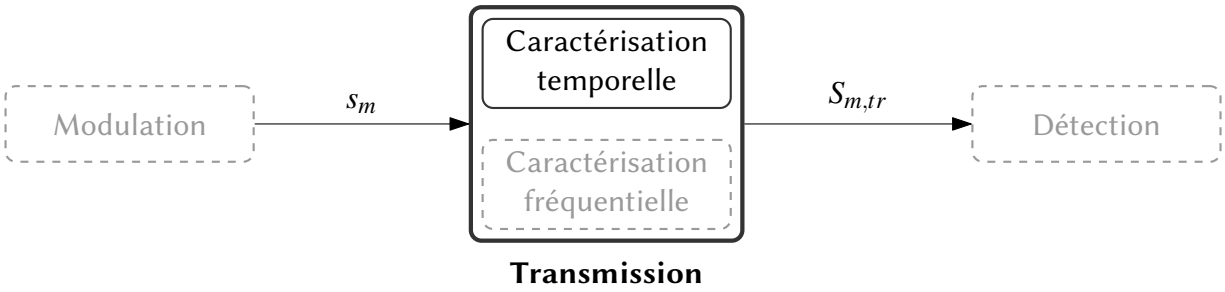
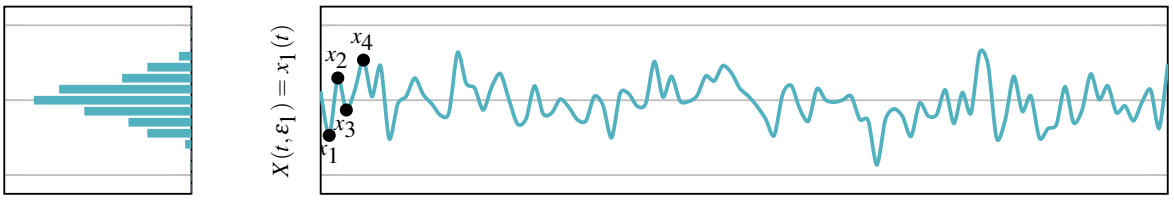


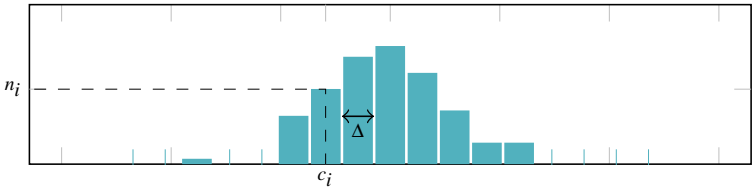
FIGURE 4 – Étape 1/2 de la caractérisation du canal de transmission : caractérisation statistique

Étude théorique de l'estimateur de la DDP

Soit X une variable aléatoire distribuée selon une certaine densité de probabilité $f_X(x)$ inconnue. On dispose de n échantillons $\{x_j\}_{j=1,\cdots,n}$ d'une réalisation de X.



On cherche à estimer empiriquement la densité de probabilité $f_X(x)$ en construisant l'histogramme des données à partir de cette réalisation. On s'intéresse donc à la construction du graphe de gauche, reproduit ci-dessous :



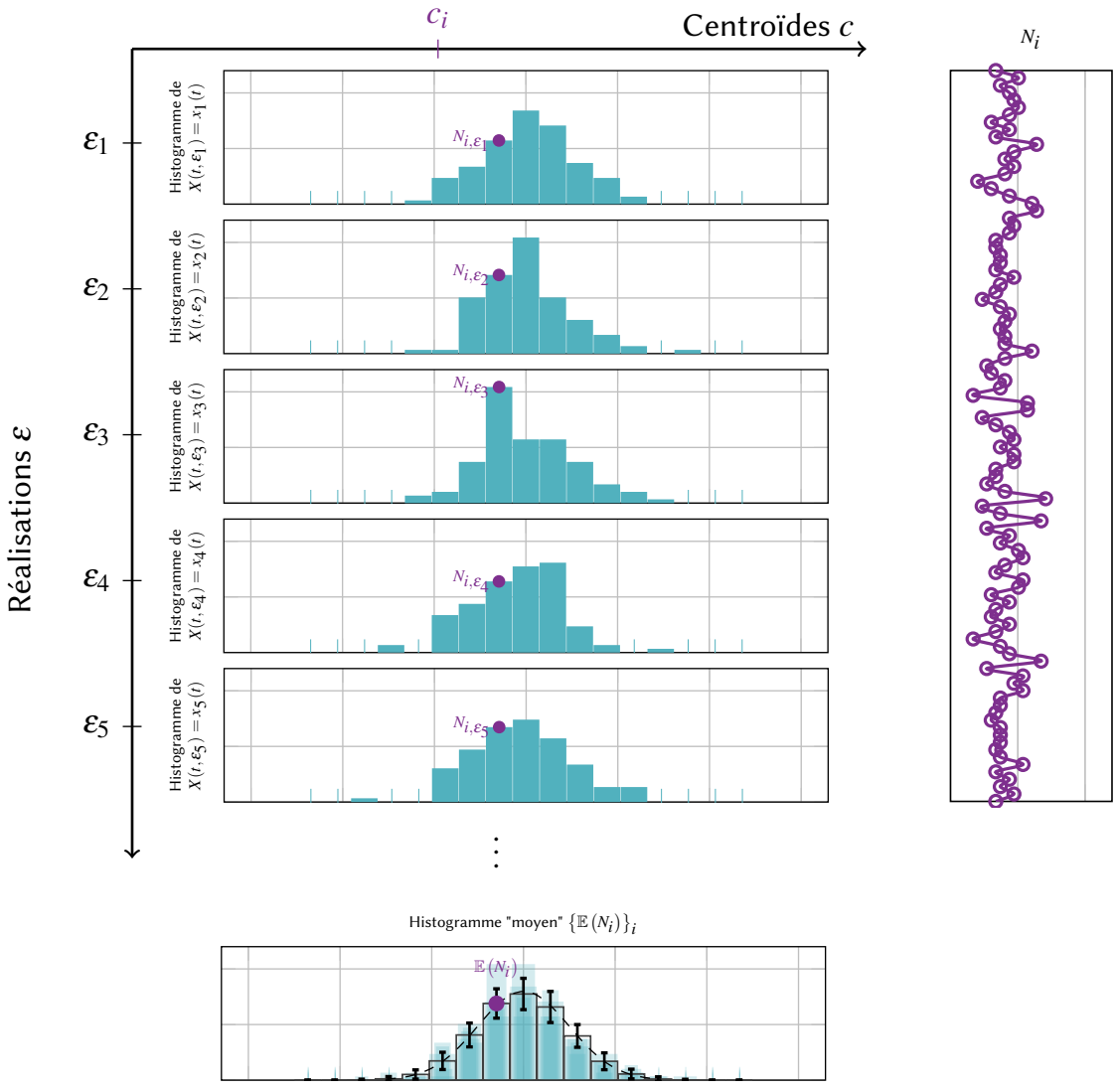
À chaque classe (de centroïde) c_i de l'histogramme de largeur Δ (avec Δ bien choisi), on associe le nombre n_i d'éléments dans cette classe :

$$n_i = \text{Card}\left\{x_j : c_i - \frac{\Delta}{2} \leq x_j < c_i + \frac{\Delta}{2}\right\}. \tag{3}$$

Ainsi, par définition des n_i, $n = \sum_i n_i$.

Q 2.1 : Calculer la probabilité $P\left(c_i - \frac{\Delta}{2} \leq X < c_i + \frac{\Delta}{2}\right)$ que la variable aléatoire X appartienne à la classe c_i en fonction de $f_X(c_i)$ et de Δ.

Q 2.2 : On effectue un grand nombre de fois l'expérience qui consiste à *i)* générer une réalisation de X (donc n échantillons), puis *ii)* construire l'histogramme des données.
Si on se concentre sur une classe c_i, pour chaque réalisation, la valeur de n_i est alors aléatoire. On définit donc la variable aléatoire N_i comme étant l'ensemble des effectifs n_i de la classe c_i (en violet dans la figure ci-dessous). Comme précédemment, une réalisation n_i de N_i est obtenue à partir de la définition (3).



Montrer que la variable aléatoire N_i suit une loi binomiale. Déterminer sa moyenne statistique et sa variance.

Q 2.3 : On considère la variable aléatoire normalisée $H_i = \dfrac{N_i}{n\Delta}$. Calculer $\mathbb{E}[H_i]$.

Q 2.4 : L'histogramme normalisé est composé de l'ensemble des bins de hauteur H_i. Que peut-on conclure du biais de l'estimation de la densité de probabilité $f_X(x)$ par l'histogramme normalisé ?

Q 2.5 : Calculer la variance $\mathrm{V}[H_i]$ de cet estimateur, puis la précision d'estimation (ou coefficient de variation), définie par

$$\varepsilon^2 = \frac{\mathrm{V}[H_i]}{\mathbb{E}[H_i]^2}.$$

En déduire, si elle existe, la valeur de Δ minimisant la précision d'estimation (i.e. correspondant au meilleur compromis biais / variance).

Analyse *in-silico* - Étude de l'estimateur de la DDP sur un cas "laboratoire"

On souhaite analyser la qualité (en terme de biais et de variance) de l'histogramme normalisé pour estimer la densité de probabilité (DDP) d'un signal aléatoire.
Étant donnée la complexité des différents canaux de transmission, il peut s'avérer difficile (voire impossible) de comparer les différents estimateurs de DDP directement en situation réelle.

Méthodologie

Validation. On se ramène en général à un cas d'étude synthétique ("exemple jouet"), dont on connaît toutes les caractéristiques. Cela nous permet alors de valider la méthode et les paramètres les plus adaptés à notre situation réelle.

Étude d'hyper-paramètres. La méthode considérée possède un certain nombre de paramètres, dont le choix pourra influer sur la qualité d'estimation (i.e., sur le biais et/ou la variance de l'estimateur). Ainsi, il s'agira dans un premier temps de faire varier **un à un** chacun de ses paramètres, et de comparer les valeurs théoriques de l'exemple jouet à la valeur expérimentale calculée.

Pour mettre en place cette méthodologie dans notre cas :

Validation. On utilisera dans un premier temps le canal de transmission test `b = canal_test()` avec `b` le signal en sortie du canal test. Les valeurs théoriques pour ce canal test sont fournies par la documentation technique de la fonction.

Étude d'hyper-paramètres. On implémentera une fonction `DDPest, c, Delta = histo(x,N=None,M=None)` prenant en entrée

- `x` : la variable contenant les données à analyser,
- `N` *(optionnel)* : le nombre d'échantillons du signal x à considérer,
- `M` *(optionnel)* : le nombre de classes imposé pour le calcul de l'histogramme.

Cette fonction doit effectuer les étapes suivantes :

- si le nombre d'échantillons N est spécifié :
  - › tronquer le signal x de 0 à $N-1$;
- si le nombre d'intervalles M n'est pas spécifié :
  - › choisir le Δ optimal vu dans l'étude théorique,
  - › calculer le nombre d'intervalles M correspondant ;
- si le nombre d'intervalles M est spécifié :
  - › déterminer la largeur des intervalles Δ correspondant à ce choix de M ;

et calculer puis retourner :

- `DDPest` : l'histogramme H_i, calculé à l'aide de la fonction `np.histogram(..., bins=... , density=False)`,
- `c` : le vecteur des centroïdes de chaque classe de l'histogramme,
- `delta` : la valeur de Δ utilisée pour le calcul de l'histogramme.

Compléter de plus la fonction d'affichage
`compareDDP(c, DDPest, DDPth, std_estim, title=None)`
qui trace sur un même graphe :

- `DDPest` : l'histogramme H_i correspondant à votre DDP estimée,
- `DDPth` : la courbe de la DDP théorique vraie $f_X(x)$ du canal test,
- ainsi que les deux courbes $f_X(x) \pm \varepsilon$ en pointillés (intervalle de précision) dépendant de l'écart-type de l'estimateur (voir étude théorique) `std_estim`,
- en fonction des centroïdes `c`.

Influence de N

On fixe le nombre d'intervalles pour le calcul des histogrammes constant et égal à $M = 20$.

Q 2.6 : Étudier la qualité de l'estimation fournie par l'histogramme normalisé en terme de biais et de variance par rapport à la DDP théorique lorsqu'on fait varier (i.e. en choisissant et en affichant le résultat pour plusieurs valeurs du paramètre) :

› le nombre d'échantillons de signal N pris en compte.

Q 2.7 : Synthétiser les analyses dans un tableau comme suit :

	biais	variance
N petit		
N grand		

TABLE 3 – Effet des différents paramètres sur la qualité de l'estimation de la DDP par l'histogramme normalisé.

Q 2.8 : Quelle propriété du signal, en général fondamentale mais difficile à vérifier en pratique, cela met-il en évidence ?

Q 2.9 : En toute rigueur, quelle expérience faudrait-il mener pour caractériser empiriquement et précisément le biais et la variance d'un estimateur ?

Pour aller plus loin : Implémenter cette procédure.

Influence de Δ

Cette fois-ci, on fixe le nombre d'échantillons à $N = 1000$ points.

Q 2.10 : En procédant de même que dans la section précédente, étudier la qualité de l'estimateur en termes de biais et de variance en analysant l'influence de Δ, c'est-à-dire :

- du nombre de classes M de l'histogramme.

  On veillera à afficher, parmi d'autres choix de M, les 2 valeurs extrêmes possibles pour M, ainsi que l'estimation pour le choix optimal de Δ.

Q 2.11 : Compléter le tableau de synthèse 3 avec ces nouvelles analyses.

Analyse *in-situ* - Caractérisation statistique du canal de transmission en situation réelle

L'étude de l'influence des paramètres de l'histogramme normalisé sur la qualité d'estimation de la DDP ayant été réalisée, on peut maintenant revenir au canal de transmission réel (pour lequel nous n'avons plus accès aux courbes théoriques !).

Pour modéliser numériquement la transmission, on dispose dans la librairie `msicpe.tsa` de la fonction `sm_tr = tsa.transmit(sm, canal_id, powB=1)`, dont les variables d'entrée sont :

- `sm` : le signal modulé en entrée du canal de transmission,
- `canal_id` : l'identifiant du canal de transmission,
- `powB` *(optionnel)* : la puissance du bruit additif du canal,

et les variables de sortie désignent :

- `sm_tr` : le signal modulé en sortie du canal de transmission

Q 2.12 : Avec des paramètres correctement choisis, caractériser statistiquement le canal de transmission :

- densité du bruit,
- moments,
- puissance du bruit.

Estimation de la Densité Spectrale de Puissance Moyenne

Objectifs pédagogiques de la séance

- Connaître un estimateur de la DSP d'un signal aléatoire
- Examiner l'influence des hyperparamètres de l'estimateur sur sa qualité (biais, variance)
- Mettre en oeuvre numériquement un estimateur de la DSP
- Déterminer le meilleur estimateur de la DSP d'un signal aléatoire
- Déterminer la DSP d'un signal aléatoire et ses caractéristiques fréquentielles associées

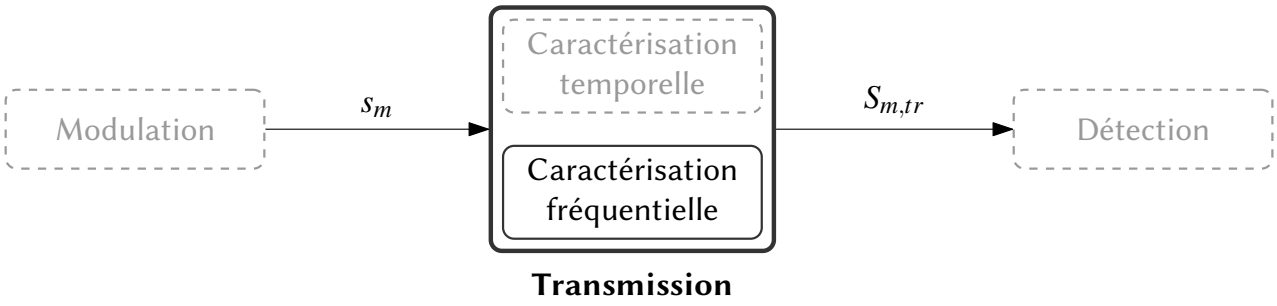


FIGURE 5 – Étape 2/2 de la caractérisation du canal de transmission : caractérisation fréquentielle

Analyse *in-silico* - Étude comparative d'estimateurs de la DSP sur un cas "laboratoire"

On souhaite analyser la qualité (en terme de biais et de variance) de trois estimateurs de la DSPM afin de sélectionner le meilleur : l'estimateur simple, l'estimateur moyenné, et l'estimateur de Welch.

En réutilisant la méthodologie employée pour l'étude de l'estimateur de la DDP :

Validation. On utilisera à nouveau le canal de transmission test `b = canal_test(filtered=True)` avec `b` le signal en sortie du canal test. Les valeurs théoriques de DSPM pour ce canal test sont fournies par la fonction `DSPth, DSPbiais = tsa.sptheo(estimateur)` dont les variables de sortie désignent :

- `DSPth` : DSPM théorique $\Gamma_X(f)$ (en dB) du canal de transmission de test,
- `DSPbiais` : DSPM théorique $\left(\Gamma_X * W_B^N\right)(f)$ (en dB) obtenue en utilisant l'estimateur `estimateur` adaptée au canal de transmission de test.

Étude d'hyper-paramètres. On implémentera pour chaque estimateur une fonction `DSPest,f = estimateur_???(...)`, dont les paramètres d'entrée seront précisés pour chaque cas.
Cette fonction doit calculer et retourner en sortie :

- `f` : le vecteur de fréquences réduites f allant de 0 à $1-\Delta f$,
- `DSPest` : l'estimation $\widehat{\Gamma}_{???}(f)$ de la DSPM en utilisant l'estimateur `???` pour la séquence considérée.

une fonction d'affichage `compareDSP(f, DSPth, DSPbiais, DSPest)` commune aux trois estimateurs, qui trace en dB sur un même graphe :

- `DSPth` : la DSPM théorique vraie du canal test $\Gamma_X(f)$,
- `DSPbiais` : la DSPM théorique obtenue en utilisant l'estimateur `???` du canal test $\left(\Gamma_X * W_B^N\right)(f)$,
- `DSPest` : votre DSPM estimée $\widehat{\Gamma}_{???}(f)$,
- en fonction du vecteur de fréquences réduites `f` tel que $0 \leq f < 0.5$.

On veillera à choisir une dynamique adaptée (par exemple, entre $+10$ dB et -50 dB).

Estimateur spectral simple

Q 3.1 : Implémenter la fonction `DSPest1,f = estimateur_simple(x, nd, N, nfft)` prenant en entrée :

- `x` : la variable contenant les données à analyser,
- `nd` : le numéro du premier échantillon de la séquence à prendre en compte,
- `N` : le nombre d'échantillons total à analyser,
- `nfft` : le nombre de points de transformée de Fourier (choisir une puissance de 2).

Q 3.2 : Quelle est la résolution spectrale de cet estimateur ?

Q 3.3 : En utilisant les fonctions développées, comparer les performances de l'estimateur simple en termes de biais et de variance aux valeurs théoriques `DSPth` et `DSPbiais` en faisant varier :

- le nombre d'échantillons de signal N pris en compte,
- la position de la réalisation considérée dans la séquence à N fixé (i.e., nd),
- le nombre $nfft$ de points utilisés pour le calcul de la transformée de Fourier.

Q 3.4 : Conclure sur la qualité de l'estimateur simple. Par exemple, il est possible de synthétiser les analyses dans un tableau comme suit :

	biais	variance
$N \nearrow$		
$nd \nearrow$		
$nfft \nearrow$		

TABLE 4 – Effet des différents paramètres sur la qualité de l'estimateur simple.

Estimateur spectral moyenné

Q 3.5 : Implémenter la fonction `DSPest2,f = estimateur_moyenne(x, M, nfft)` dont les paramètres d'entrée sont :

- `x` : la variable contenant les données à analyser,
- `M` : le nombre d'échantillon par segment,
- `nfft` : le nombre de points de transformée de Fourier (choisir une puissance de 2).

On utilisera la fonction `scipy.signal.welch` correctement paramétrée pour effectuer l'estimation moyennée.

Q 3.6 : Quelle est la résolution spectrale de cet estimateur ?

Q 3.7 : Étudier et conclure quant aux performances de l'estimateur moyenné à $N \approx 10000$ points fixé, en fonction :

› de la taille M des segments et (donc de leur nombre : $L = N/M$).

Q 3.8 : Choisir un nombre de segments M^* conduisant à un biais « acceptable » jusqu'à -30 dB. Pour ce choix, mesurer la variabilité de l'estimation dans la bande passante du bruit (on pourra par exemple mesurer l'écart-type).

Estimateur spectral de Welch

Q 3.9 : Implémenter la fonction
`DSPest3,f = estimateur_welch(x, window, M, Noverlap, nfft)`
dont les paramètres d'entrée sont :

- `x` : la variable contenant les données à analyser,
- `window` : le nom de la fenêtre sous la forme d'une chaîne de caractères (voir la doc de `get_window()`),
- `M` : le nombre d'échantillon par segment,
- `Noverlap` : le nombre de points de recouvrement,
- `nfft` : le nombre de points de transformée de Fourier (choisir une puissance de 2).

On utilisera la fonction `scipy.signal.welch` correctement paramétrée pour effectuer l'estimation.

Q 3.10 : Quelle est la résolution spectrale de cet estimateur ?

Q 3.11 : Avec la valeur de M^* retenue pour l'estimateur moyenné, et en choisissant une fenêtre rectangulaire, étudier l'influence du taux de recouvrement (25%, 50% et 75%), i.e., l'impact du choix de $N_{overlap}$, sur les performances de l'estimateur de Welch en termes de biais et de variance.

› Comment se comporte le biais d'estimation en fonction du taux de recouvrement ? Pourquoi ?

› Comparer ces valeurs à celle mesurée pour l'estimateur moyenné.

› Expliquer ces résultats en les confrontant à l'étude théorique vue en cours.

› Que permet le recouvrement entre segments ?

Q 3.12 : En conservant le choix de M^*, étudier l'influence du choix de la fenêtre sur le biais et la variance de l'estimateur de Welch.

› Lancer la fonction `tsa.fenetre()`. Étudier les propriétés des différentes fenêtres disponibles, en particulier :

- la largeur du lobe principal,
- la largeur des lobes secondaires,
- le taux d'atténuation d'amplitude des lobes secondaires.

Sur quelle(s) caractéristique(s) de l'estimation de la DSPM ces fenêtres vont-elles agir ?
On pourra par exemple indiquer quel serait le profil de la fenêtre idéale.

› Que peut-on dire de la variance d'estimation pour chacune de ces fenêtres ? Pourquoi ?

Q 3.13 : Conclure sur l'influence des paramètres et la qualité de l'estimateur de Welch, en s'inspirant par exemple du tableau 4.

Q 3.14 : Pour quel(s) paramétrage(s) de l'estimateur de Welch (M, recouvrement, fenêtre) obtient-on des résultats satisfaisants ?

Synthèse

Q 3.15 : En expliquant la démarche scientifique suivie, sélectionner parmi les trois estimateurs étudiés précédemment celui qui semble le plus adapté. Synthétiser les résultats dans un tableau.

Analyse *in-situ* - Caractérisation spectrale du canal de transmission en situation réelle

L'étude comparative des trois estimateurs de la DSP ayant été réalisée, on peut maintenant revenir au canal de transmission réel (pour lequel nous n'avons plus accès aux courbes théoriques !).

Q 3.16 : Avec l'estimateur choisi à la **Q 3.15**, réaliser l'analyse spectrale du canal de transmission :

› la bande passante du bruit généré par le canal de transmission,
› la nature de ce bruit,
› la puissance moyenne de ce bruit.

Détection d'un signal noyé dans un bruit

Objectifs pédagogiques de la séance

- Comprendre et implémenter un filtrage adapté pour la détection
- Examiner l'influence des paramètres externes sur la qualité de la détection
- Différencier filtrage adapté et filtrage optimal
- Mesurer les performances d'un filtre dans une tâche de débruitage
- Décoder un signal binaire vers un type de données prescrit

Une fois le signal modulé, il est transmis jusqu'au récepteur. Le signal $S_{m,tr}$ récupéré en sortie est fortement dégradé : le signal utile s_m est noyé dans le bruit du canal de transmission, dont les caractéristiques temporelles (DDP, moments) et fréquentielles (DSP) ont été étudiées précédemment. Il s'agit donc maintenant de détecter la présence de signal dans le bruit, en mettant en place un filtrage adapté. Puis de décoder le signal originalement transmis.

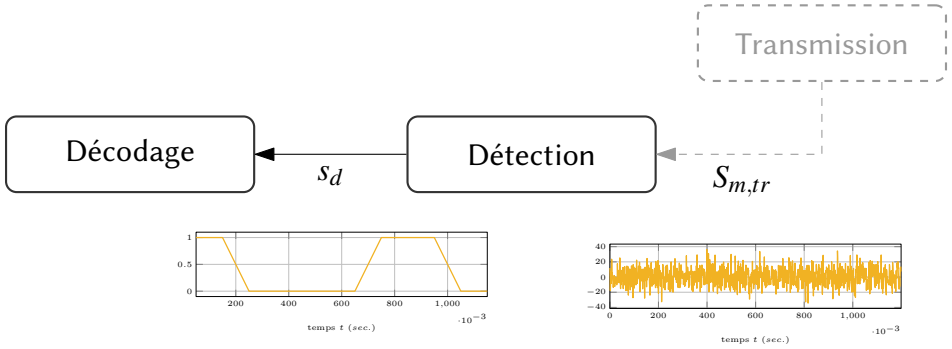


FIGURE 6 – Étape de détection du signal noyé dans un bruit de caractéristiques connues

Étude théorique de la détection et de la reconstruction

Détection par filtrage adapté

Étant donné que le signal utile s_m est noyé dans du bruit, on va procéder à la détection de chaque symbole '0' ou '1' par filtrage adapté.

Q 4.1 : Donner le schéma de principe de la détection de chaque symbole en précisant l'expression mathématique des réponses impulsionnelles $h_0(t)$ et $h_1(t)$ des filtres correspondants. Que doit-on obtenir en sortie de chacun des filtres adaptés ?

Q 4.2 : On combine les sorties pour récupérer un seul signal. Quel effet la détection a-t-elle finalement eu sur le signal $S_{m,tr}$?

Performance de détection

Q 4.3 : Quel critère permet de quantifier la qualité d'un signal bruité ? Rappeler son nom et sa définition générale.

Q 4.4 : Quelle caractéristique du bruit a une influence sur ce critère ?

Q 4.5 : Exprimer ce critère avant filtrage en fonction de l'amplitude A des signaux utiles et de la DSPM du bruit.

Q 4.6 : Exprimer la puissance du signal et la puissance du bruit après filtrage (on pourra éventuellement s'appuyer sur des schémas). En déduire l'expression de ce critère après filtrage.

Décodage du signal

Le décodage consiste à convertir un nombre binaire en sa représentation décimale. C'est l'opération inverse de la binarisation.

Q 4.7 : Quel *a priori* est-il nécessaire pour décoder correctement le signal ?

Débruitage du signal transmis par filtrage adapté

Transmission

Maintenant qu'on a pu caractériser la Densité de Probabilités (DDP) et la Densité Spectrale de Puissance Moyenne (DSPM) du bruit additif du canal de transmission, il est temps de transmettre dans ce canal de transmission le signal modulé s_m.

Construire le signal $S_{m,tr}$ récupéré en sortie de canal de transmission.

Débruitage par détection

Implémenter la détection des symboles '0' et '1' dans le signal $S_{m,tr}$ via les deux filtres adaptés de réponses impulsionnelles h_0 et h_1 exhibés dans l'étude théorique.

Afin de reconstruire la séquence binaire s_d, il faudra comparer la sortie des deux filtres adaptés au centre de chaque période T_{mod}. Pour cela, afficher sur une même figure :

- le signal binaire initial s_b,
- le signal modulé s_m,
- la sortie de h_0 au signal $S_{m,tr}$ transmis,
- la sortie de h_1 au signal $S_{m,tr}$ transmis.

En déduire le signal s_d résultant.

Étude de l'effet du bruit de transmission sur la précision

Afin d'évaluer les performances de cette approche de débruitage par détection, nous spécifierons dans cette partie le troisième paramètre `noisePow` à la fonction `tsa.transmit`, qui détermine la puissance du bruit introduit par le canal de transmission.

Implémenter une fonction `e = error(s_d, s_b)`, qui prend en entrée :

- le signal détecté `s_d`,
- le signal binaire original `s_b`,

et qui renvoie `e` le taux (%) de mauvaise détection (faux positifs et faux négatifs).

Q 4.8 : Tracer l'évolution du taux de mauvaise détection entre les signaux s_b et s_d en fonction du SNR pour différentes puissances de bruit. Que remarquez-vous ?

Q 4.9 : Cela est-il concordant avec les notions vu en cours ?

Décodage

Décoder le signal binaire s_d en utilisant la fonction `bin2data`.

Prédiction AR d'ordre M

Objectifs pédagogiques de la séance

- Énoncer les hypothèses qui sous-tendent le principe de prédiction linéaire basée sur un modèle auto-régressif
- Établir un modèle auto-régressif (AR)
- Appliquer le principe de prédiction linéaire basée sur un modèle auto-régressif au cas du débruitage d'un signal

En sortie de décodage, on se rend compte que le signal obtenu est dégradé par un bruit impulsionnel, qui peut être dû soit à la défaillance à certains instants du capteur (saturation), soit à des erreurs lors de l'étape de décodage (erreur sur des bits de poids forts). On se propose donc dans une dernière étape d'améliorer la qualité du signal reçu par filtrage optimal, afin d'éliminer le bruit.

FIGURE 7 – Étape de débruitage du signal reçu

Q 5.1 : Quel type de bruit semble présent dans le signal résultant du décodage ?

Q 5.2 : Quelle(s) solution(s) simple(s) pourrait-on envisager pour supprimer (ou atténuer) l'effet de ces pics ? En quoi ne seraient-elles pas satisfaisantes ?

La solution retenue ici, consiste à modéliser le signal obtenu par un processus auto-régressif d'ordre M (AR(M)). Ce modèle est ensuite utilisé pour prédire l'échantillon $s_{tr}[n]$ du signal à partir des M échantillons précédents $\{s_{tr}[n-m], 1 \leq m \leq M)\}$. Puisque la dégradation est aléatoire, et supposée indépendante du signal, le modèle AR(M) n'aura pas le pouvoir de les prédire. L'analyse de l'erreur de prédiction pourra donc nous permettre de conclure.

Étude théorique des coefficients du filtre AR prédictif

Soient $\{S[n]\}_{n=1,\ldots,N}$, les échantillons d'un signal aléatoire réel, stationnaire. On se propose de trouver un modèle tel que l'on puisse prédire linéairement la valeur de $S[n]$ à partir des échantillons précédents de S :

$$\hat{S}[n] = \sum_{k=1}^{+\infty} h[k]S[n-k].$$

Il s'agit donc de déterminer les coefficients $h[k]$, $k = 0, 1, \cdots$ minimisant l'erreur de prédiction :

$$\varepsilon[n] = S[n] - \hat{S}[n] = S[n] - \sum_{k=1}^{+\infty} h[k]S[n-k] \tag{4}$$

au sens de l'erreur quadratique moyenne (i.e. puissance moyenne minimale) $P_\varepsilon = \mathbb{E}\left[\varepsilon[n]^2\right] = \sigma^2$.

Q 5.1 : Quel principe de construction permet de garantir une erreur quadratique moyenne minimale ?

Q 5.2 : Par application de ce principe, montrer que l'erreur de prédiction ε est un bruit blanc.

Q 5.3 : En pratique, il faut limiter l'ordre du modèle à $M < +\infty$. Dans ces conditions et toujours par application de ce même principe de construction, établir le système d'équations linéaires, dont les coefficients $\{h[k], k = 1, \cdots, M\}$ sont les solutions.

Q 5.4 : Calculer la puissance de l'erreur de prédiction P_ε en fonction de l'autocorrélation $\gamma_s(k)$ du signal et des coefficients $\{h[k], k = 1, \cdots, M\}$.

Q 5.5 : Insérer cette relation dans le système d'équations linéaires obtenu à la question 3.

Q 5.6 : En déduire l'expression des coefficients $\{h[k], k = 1, \cdots, M\}$ du filtre prédictif.

Implémentation d'un filtre prédictif AR d'ordre M

Estimation de la fonction d'autocorrélation

Estimer la fonction d'autocorrélation $\gamma_{s_{tr}}[k] = \gamma_{s_{tr}}(kT_s)$ du signal transmis s_{tr}, pour $-K \leq k \leq K$. Pour cela, on utilisera le bloc d'instructions suivant :

```
Gam = signal.correlate(s_tr.astype(np.float32),
 ↪  s_tr.astype(np.float32), mode='full', method='auto')
Gam /= len(s_tr)
Gam = Gam[len(Gam)//2 - K:len(Gam)//2 + K + 1]
lags = signal.correlation_lags(len(s_tr), len(s_tr),
 ↪  mode="full")
lags = lags[len(lags)//2 - K:len(lags)//2 + K + 1]
```

où

- `s_tr` est le signal à corréler
- `Gam` est le vecteur des corrélations
- `lags` est le vecteur des retards $-K \leq k \leq K$

Afficher le résultat pour $K = 200$.

Q 5.7 : À partir de cette fonction, justifier le choix de l'ordre du modèle AR(M).

Identification du modèle AR(M)

Calculer numériquement les coefficients $\{h[k], k = 1, \cdots, M\}$ solutions du système linéaire exhibé analytiquement. On utilisera pour cela, la fonction `toeplitz` de la librairie `scipy.linalg`.
Note : $\gamma_{s_{tr}}$ est la fonction d'autocorrélation empirique estimée à partir des échantillons de s_{tr}. La matrice de Toeplitz peut donc ne pas être inversible. On utilisera alors la commande

`numpy.linalg.pinv` pour calculer la matrice pseudo-inverse (de Moore-Penrose) qui, elle, existe toujours.

Afficher les coefficients $\{h[k], k = 1, \cdots, M\}$ du modèle AR(M) ainsi obtenus.

Prédiction linéaire

En utilisant les coefficients $\{h[k], k = 1, \cdots, M\}$ du modèle AR(M) ainsi identifié, calculer la prédiction à un pas de temps $\hat{s}_{tr}[n]$ de $s_{tr}[n]$ à partir des échantillons précédents $\{s_{tr}[n-k], k = 1, \ldots, M\}$, pour $n \geq M+1$.

Afficher la prédiction du signal ainsi obtenue superposée au signal original.

Afficher dans une autre figure la valeur absolue de l'erreur de prédiction $\varepsilon[n] = \hat{s}_{tr}[n] - s_{tr}[n]$.

Q 5.8 : Choisir un seuil σ pour identifier, à partir de l'erreur de prédiction, les indices k_i des dates de changement de comportement (dus au bruit impulsionnel).

Restauration par prédiction causale / anticausale. La prédiction causale (estimation de l'échantillon $s[n]$ à partir des échantillons antérieurs) produit une erreur de prédiction où chaque changement est repéré par un pic principal, suivi d'un train de pics secondaires (rebonds dus au temps de réponse du filtre $h[k]$). Cela rend donc la détection des changements brusques dans le signal moins précise.

Pour éviter cela, on peut donc combiner à la prédiction causale déjà réalisée une prédiction anticausale, où chaque échantillon s_n est prédit à partir des M échantillons suivants $\{s[n+k], k = 1, \cdots, M\}$. Cela peut se faire en utilisant le même filtre $h[k]$ puisque la fonction d'autocorrélation est paire. Les erreurs de prédiction $\varepsilon_{\text{causale}}$ et $\varepsilon_{\text{anticausale}}$ n'ont alors en commun que les pics principaux localisés aux instants des comportements qu'on souhaite.

Implémenter la prédiction anti-causale, et en déduire une analyse plus précise du signal.