École Nationale
Supérieure
de l'Électronique
et de ses Applications

Modélisation des signaux aléatoires
Majeure de Signal

Inbar Fijalkow, Matthieu Guerquin-Kern [1]

É.N.S.É.A. 2023–2024, 2e année

Projet

Les quatre séances de TP de majeure de Signal du semestre 7 (« Modélisation des Signaux Aléatoires ») prennent la forme d'un projet unique réalisé dans l'environnement Matlab/Octave. Les fichiers nécessaires sont accessibles sur Moodle.

L'évaluation se fera sur la base du travail en séance et d'un compte-rendu détaillé délivré à la fin de la dernière séance. Vous devez valider **des check-points pendant chaque séance**, avec le.a professeur.e encadrant qui les notera. **A la fin de chaque séance** et avant de quitter le poste de travail, chaque binôme a l'obligation de **charger sur Moodle un fichier .zip contenant le code Matlab** pour cette séance.

Le compte-rendu final devra comporter :

— un détail des voies explorées au cours du projet,
— une description des méthodes mises en place pour vérifier la validité du code produit et des résultats obtenus.

1. Avec modification mineures de Laura Luzzi.

Principales fonctions Matlab/Octave utiles

`randn`	génération d'échantillons indépendants, gaussiens, centrés, de variance 1.
`rand`	génération d'échantillons indépendants suivant une loi uniforme continue sur $[0, 1]$.
`plot`	tracé de graphes
`subplot`	découpage de figure en plusieurs graphes.
`filter`	filtrage d'un signal par un filtre ARMA (défini par son équation aux différences).
`load`	lecture de fichiers stockant des variables au format `.mat`.
`max`	trouver le maximum d'un vecteur et son indice.
`fft`	calcul d'une transformée de Fourier discrète (attention à la normalisation).
`audioread`	récupération d'un signal à partir d'un fichier son.
`audiowrite`	écriture d'un signal dans un fichier son.
`soundsc`	écoute d'un signal sonore.

Pour avoir des détails sur une fonction Matlab, par exemple `randn`, utiliser `doc randn` ou `help randn`.

Contexte du projet : analyse et synthèse vocale

L'objectif de ce projet est de mettre en œuvre une chaîne de codage et décodage de la voix humaine utilisant le *linear predictive coding* (codage par prédition linéaire). Une telle chaîne se retrouve dans des applications de transmission de la parole économes en bande passante (voir normes GSM 06.10 ou LPC-10/FS-1015), où l'objectif est d'extraire les caractéristiques minimales du signal à transmettre (étape d'analyse du codeur) et d'être capable, à la réception, de synthétiser (codeur) un signal fidèle à l'original.
La maîtrise des étapes d'analyse et de synthèse ouvre également la voie à des applications créatives où un effet est appliqué à la parole en suivant trois étapes :

- Analyse des caractéristiques de la parole.
- Manipulation de ces caractéristiques pour obtenir un effet désiré (par exemple, changement de fréquence).
- Synthèse de la parole à partir des caractéristiques modifiées.

On comparera les signaux réels et synthétiques avec les outils de traitement du signal (autocorrélation, spectre) et à l'oreille.

1 Un modèle source/filtre pour la voix

Des modèles plus ou moins complexes des mécanismes de production de la parole peuvent être établis.

Le plus simple d'entre eux, illustré en figure 1, consiste à considérer que le signal représentant la parole est issu d'un filtre linéaire, modélisant le conduit vocal, qui est excité par une source indépendante, modélisant l'expiration passant à travers la glotte. Cette source glottique peut prendre deux formes :

voisée correspondant à des vibrations vocales, modélisée par une impulsion périodique à phase aléatoire (c'est par exemple le cas pour les voyelles, ainsi que de 'b', 'd' et 'z'),

non voisée correspondant à un écoulement turbulent du flux d'air expiré, modélisé par un bruit blanc (c'est par exemple le cas pour 'p', 't', 's' et 'ch').

Cette modélisation est particulièrement utile pour élaborer des outils de compression de la parole : la parole n'est plus représentée comme une suite d'échantillons, mais directement à partir d'une suite de modèles source/filtre. Dans ce contexte, la chaîne de traitement de la parole comprend deux étapes :

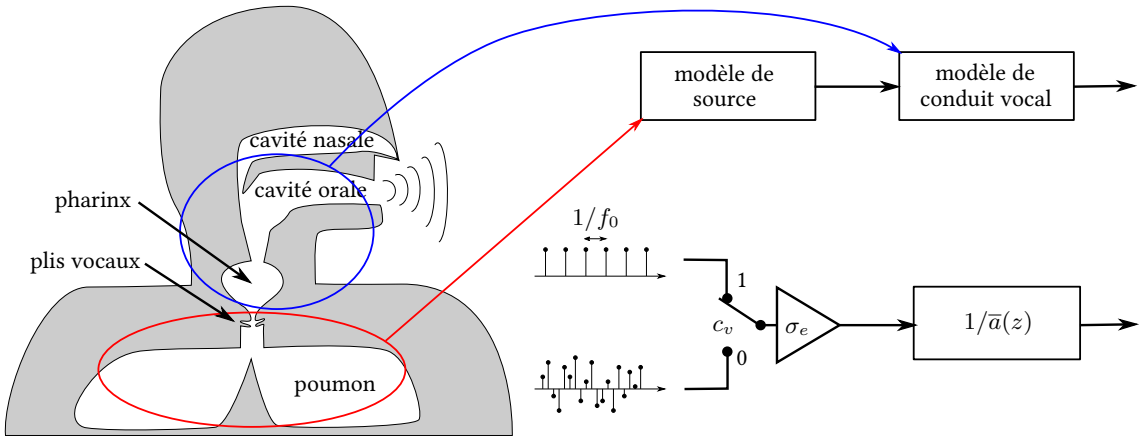


FIGURE 1 – Illustration du modèle source/filtre.

— une partie *codage* (*analyse*) dont le but est d'analyser le signal de parole pour en déduire une représentation sous forme de modèles source/filtre dont les paramètres sont transmis. Ainsi le débit d'information est réduit (compression).
— une partie *décodage* (*synthèse*) permettant de reconstruire le signal de parole à partir de la seule donnée des modèles source/filtre.

Pour en savoir plus sur l'anatomie de l'appareil vocal et sa modélisation, on peut se rapporter respectivement aux chapitres 1 et 2 de la thèse de Thomas Hézard (2014) :
`https://hal.archives-ouvertes.fr/tel-00933070/document`.

2 Une stationnarité limitée dans le temps

Le modèle source/filtre est satisfaisant pour expliquer l'origine des sons constituant la parole (voir figure 2). Alors, on peut établir des modèles aléatoires stationnaires aux ordres 1 et 2 selon que le son est voisé ou non. Néanmoins, au cours du temps, les caractéristiques du modèle sont amenées à évoluer, ce qui rompt la stationnarité.

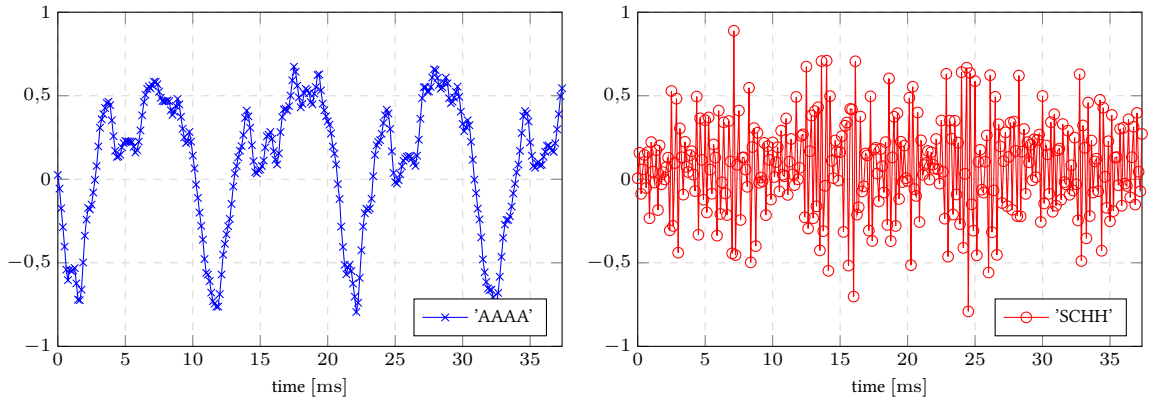


Figure 2 – Évolutions temporelles de son voisé (à gauche) et non voisé (à droite). Le son voisé présente une pseudo-périodicité, il est très structuré. On le modélise bien avec une impulsion périodique filtrée. À l'inverse, le son non voisé est peu structuré (régulier) : il se modélise bien avec un bruit blanc filtré.

On considère généralement que pour des périodes de l'ordre de 20 à 30 ms, l'hypothèse de stationnarité du signal de parole est satisfaite.

Dans le cadre de ce projet, nous vous proposons de travailler avec une fréquence d'échantillonnage F_e de 10 kHz, suffisante pour le signal de parole, en analysant le signal par tronçons de 256 échantillons.

2.1 Schéma d'analyse et de synthèse à suivre

Dans ce projet, nous vous proposons de suivre le schéma présenté en figure 3. Il se décompose en deux étapes.

L'analyse Cette première étape de la chaîne de traitement du signal de parole a pour buts :
— de découper le signal de parole en tronçons d'analyse de $N = 256$ échantillons chacune,
— de déterminer, pour chaque tronçon, le filtre prédicteur optimal de M paramètres $\mathbf{a}_{\text{opt}}$, le caractère voisé ou non c_v (booléen) du tronçon et les caractéristiques du signal d'excitation : variance du signal d'excitation σ_e^2 et fréquence fondamentale f_0 (*pitch*) de la série d'impulsions si le son est voisé. Si le son est non voisé, le signal d'excitation, modélisé

par un bruit blanc gaussien, pourra être correctement synthétisé par la seule donnée de la variance σ_e^2.

— le codage de chaque tronçon se résume donc à $M + 3$ paramètres (au lieu de $N = 256$ échantillons) : les M coefficients de $\mathbf{a}_{\text{opt}}$, c_v , σ_e et f_0.

La synthèse La seconde étape de la chaîne de traitement consiste à reconstituer le signal de parole tronçon par tronçon : connaissant pour chaque tronçon le type d'excitation, sa puissance, les coefficients du filtre AR(M), il suffit alors de générer une excitation et d'effectuer l'opération de filtrage idoine.

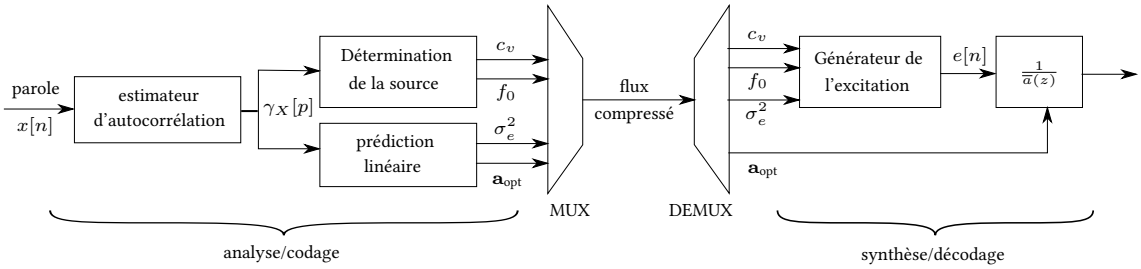


FIGURE 3 – Codage / Décodage du signal de parole.

S1 Estimation de l'autocorrélation

Objectifs de cette première séance : découverte du sujet du projet, estimation fiable de l'autocorrélation, critère pour déterminer le caractère voisé d'un son.

1 Données sur lesquelles travailler

Dans un premier temps, il peut être intéressant de tester vos estimateurs à l'aide de réalisations de processus dont vous connaissez l'autocorrélation : bruit blanc (voir `rand` ou `randn`), AR1 (voir `filter`), sinusoïde à phase aléatoire (voir `sin`).

Dans un second temps, on vous fournit une archive contenant plusieurs fichiers au format `flac` correspondant à des sons de nature voisée et non voisée. Pour récupérer les signaux et fréquences d'échantillonnage correspondant à ces fichiers, voir la fonction `audioread`.

2 Estimateur biaisé d'autocorrélation

Écrire la fonction décrite de la manière suivante :

```
%% [Cx,p] = BiasedCrossCorr(X,pmax)
% Computation of the biased cross correlation of a given signal X.
%
% INPUTS
% - X       vector of signal samples
% - pmax    maximal amount of shift to be considered
%           (optional, defaults to the length of X minus 1)
% OUTPUTS
% - Cx      vector of cross correlation samples for shifts varying
%           from 0 to pmax
% - p       vector of corresponding shifts
function [Cx,p] = BiasedCrossCorr(X,pmax)
```

Tester la fonction pour les signaux suivants :

◇ **Checkpoint** 1: réalisation d'un bruit blanc de variance donnée.

◇ **Checkpoint** 2: réalisation d'un processus AR1 avec un paramètre a et une puissance d'innovation σ^2 donnés.

◇ **Checkpoint** 3: réalisation d'une sinusoïde à phase aléatoire de fréquence réduite et d'amplitude données.

Liste non exhaustive de questions que l'on peut se poser :

- Pour des processus dont on connait l'autocorrélation théorique, cet estimateur biaisé semble-t-il converger quand le nombre d'échantillons du signal devient grand ?
- Peut-on observer l'effet du biais avec certains exemples ?
- La valeur d'autocorrélation estimée prise pour un décalage $p = 0$ est-elle en accord avec ce que renvoie la fonction `var` ?
- Cet estimateur biaisé est-il plutôt adéquat pour les sons voisés ou pour les sons non voisés ?
- Que se passe-t-il si le signal étudié contient une composante continue, traduisant un processus non centré ? Comment estimer correctement son autocorrélation ?

3 Estimateur non biaisé d'autocorrélation

Écrire la fonction décrite de la manière suivante :

```
%% [Cx,p] = UnbiasedCrossCorr(X,pmax)
% Computation of the unbiased cross correlation of a given signal X.
%
% INPUTS
% - X      vector of signal samples
% - pmax   maximal amount of shift to be considered
%          (optional, defaults to the length of X minus 1)
% OUTPUTS
% - Cx     vector of cross correlation samples for shifts varying
%          from 0 to pmax
% - p      vector of corresponding shifts
function [Cx,p] = UnbiasedCrossCorr(X,pmax)
```

◇ **Checkpoint** 4: Tester votre fonction pour le bruit blanc, le processus AR1 et la sinusoïde de la section précédente et comparer les résultats.

Liste non exhaustive de questions que l'on peut se poser :

— Pour des processus dont on connaît l'autocorrélation théorique, cet estimateur non biaisé semble-t-il converger quand le nombre d'échantillons du signal devient grand ? Observe-t-on effectivement un biais nul ?

— Peut-on observer l'effet de la variance de l'estimateur avec certains exemples ?

— Cet estimateur non biaisé est-il plutôt adéquat pour les sons voisés ou pour les sons non voisés ?

4 Établissement d'un critère pour le caractère voisé

1. Les sons voisés issus de la parole présentent une fréquence fondamentale qui varie entre 100 Hz (homme) et 400 Hz (enfant). En déduire la plage de pseudo-périodes observables dans les sons voisés.
2. En effectuant une observation temporelle des sons voisés, mesurer leur pseudo-période.
3. Retrouver ces pseudo-périodes en analysant les autocorrélations estimées.
4. Observer avec quelle rapidité la « structure » des signaux non voisés impose une décroissance de l'autocorrélation pour les décalages croissants.
5. Établir un critère permettant de distinguer les sons voisés des sons non voisés et l'implémenter.

```
%% bool = isVoiced(Cx)
% Determines if a sound is voiced based on its cross correlation values.
%
% INPUT
% - Cx     vector of cross correlation samples for shifts varying
%          from 0 to numel(Cx)-1.
% OUTPUT
% - bool   1 (true) if the sound is voiced, 0 (false) otherwise.
function bool = isVoiced(Cx)
```

5 Pour aller plus loin

— Adaptation des estimateurs pour des signaux à valeurs complexes.

— Pour les cas où l'on s'intéresse à un décalage p variant entre 0 et $p_{\max} > \log(N)$, on peut utiliser une implémentation de complexité $\mathcal{O}(N \log N)$ à l'aide de l'algorithme FFT, plus favorable que l'implémentation classique de complexité $\mathcal{O}(M p_{\max})$. Observation du gain algorithmique avec le temps d'exécution.

S2 Analyse Spectrale

Objectifs de cette deuxième séance : estimation de la densité spectrale de puissance (DSP) d'un signal.

1. Caractérisation d'un processus aléatoire dans le domaine fréquentiel,
2. Effet du fenêtrage.

1 Données sur lesquelles travailler

Comme lors de la première séance, il peut être intéressant de commencer par tester vos estimateurs à l'aide de réalisations de processus dont vous connaissez les caractéristiques spectrales : bruit blanc, AR1 *et surtout* sinusoïde à phase aléatoire.

Dans un second temps, vous pouvez pousser l'étude vers l'analyse du contenu fréquentiel des sons voisés et non voisés de l'archive fournie.

2 Deux approches pour un même estimateur (ou presque. . .)

Pour estimer la densité spectrale de puissance (DSP) d'un processus X dont on a mesuré une trajectoire sur un nombre fini N d'échantillons, deux approches sont possibles :

- **corrélogramme** : utiliser l'estimateur biaisé d'autocorrélation [2] et calculer sa transformée de Fourier,
- **périodogramme** : calculer la TFTD des échantillons dont on dispose, prendre la valeur absolue au carré et diviser par le nombre d'échantillons.

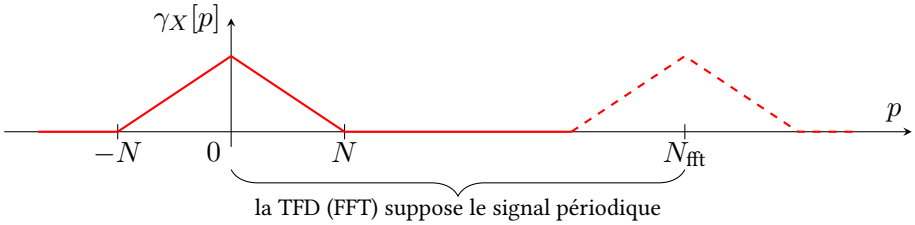
Dans la pratique, on ne peut pas estimer (ou simplement avoir accès à) la densité spectrale de puissance pour toutes les valeurs de fréquence réduite (la variable est continue). On s'intéresse donc à la DSP pour un ensemble fini de valeurs de fréquences réduites. Il est intéressant de tirer parti de l'algorithme FFT pour les calculs de transformées de Fourier discrètes, avec un paramètre N_{fft} à choisir.

Écrire la fonction réalisant l'estimation de DSP décrite de la manière suivante :

```
%% [PSD,nu] = psdEstimator(X,Nfft)
% Power spectral density estimator.
%
% INPUTS
% - X      vector of signal samples
% - Nfft   Number of signal samples to be considered in the discrete
%          Fourier transform (see FFT)
%          (optional argument, defaults to twice the length of X)
% OUTPUTS
% - PSD    vector of PSD estimates for normalized frequency varying
%          from 0 to 1/2.
% - nu     vector of corresponding normalized frequency values
function [PSD,nu] = psdEstimator(X,Nfft)
```

Remarques : On utilise l'autocorrélation estimée, qui présente une symétrie hermitienne $\gamma_X[-p] = \gamma_X[p]^*$ et possède $2N-1$ valeurs ($0 \leq |p| < N$). Quand on calcule la TFD de ces valeurs sur N_{fft} points, on suppose implicitement qu'elles constituent le motif d'un signal N_{fft}-périodique. Pour que cette périodicité ne pose pas de problème, il faut $N_{\text{fft}} \geq 2N-1$.

2. avec l'estimateur non biaisé ça ne fonctionne pas bien, vous pouvez tenter…



Pour la méthode du corrélogramme, on peut soit utiliser un décalage circulaire de manière à définir le motif sur l'intervalle $[0, N_{\text{fft}} - 1]$ (voir figure ci-dessous), soit commencer par déterminer $C[k] = \sum_{p=0}^{N-1} \gamma_X[p] \mathrm{e}^{-2\mathrm{j}\pi kp/N_{\text{fft}}}$ des seules valeurs d'autocorrélation pour un décalage $0 \leq p < N$, puis calculer $2\mathrm{Re}(C[k]) - \gamma_X[0]$ (cherchez à comprendre pourquoi !).

Avec la méthode du périodogramme, on est également censés obtenir les coefficients de la TFD sur N_{fft} points des $2N - 1$ valeurs $\gamma_X[p]$.

◇ **Checkpoint** 5: Tester votre fonction pour un bruit blanc, une sinusoïde à phase aléatoire et un processus AR1.

Liste non exhaustive de questions que l'on peut se poser :

- Quelle est la valeur minimale de N_{fft} telle que l'algorithme FFT donne des valeurs correspondant à la TFD attendue ?
- Pour la méthode du corrélogramme, quelles valeurs de décalage prendre en compte ?
- Pour la méthode du corrélogramme, comment organiser le vecteur de coefficients d'autocorrélation pour que les coefficients calculés par la FFT renvoient une valeur réelle pure (et accessoirement positive) ?
- Pour la méthode du périodogramme, par quel nombre d'échantillons normaliser les résultats ? N ou N_{fft} ?
- Quelles sont les fréquences réduites correspondant aux coefficients renvoyés par l'algorithme FFT ?
- Comment réduire les échantillons du spectre obtenus aux seuls correspondant à une fréquence réduite entre 0 et 1/2 ?
- Testez votre fonction avec des réalisations de sinusoïde à phase aléatoire dont la fréquence ν_0 est un nombre irrationnel (par exemple $\nu_0 = \sqrt{2}/8$).
- Testez la capacité de résolution de votre estimateur pour une superposition de deux sinusoïdes de fréquences ν_1 et ν_2 très proches.

3 Fiabilité de l'estimation spectrale

Grâce au théorème de Wiener-Kintchine, on sait que l'estimateur de DSP est asymptotiquement non biaisé, c'est-à-dire qu'à mesure que le nombre d'échantillons N croît, son espérance s'approche de la véritable DSP.

Néanmoins, à moins que le processus étudié ne soit totalement prédictible [3], a variance de l'estimateur de DSP ne converge pas vers zéro quand le nombre d'échantillons croît (estimateur non convergent en moyenne quadratique et donc asymptotiquement inexact).

La seule possibilité pour réduire la variance est alors de tronquer les N échantillons dont on dispose en parts avec lesquelles on estime la DSP et, finalement, moyenner les DSP obtenues. Avec K parties décorrélées, on divise alors la variance par K.

◇ **Checkpoint** 6: Adapter la fonction `psdEstimator()` pour qu'elle prenne K en troisième argument optionnel et réalise un moyennage le cas échéant.

Liste non exhaustive de questions que l'on peut se poser :

3. très forte structure se traduisant par une absence de décroissance de l'autocorrélation avec le décalage ou par la présence de raies dans la DSP

— Comment la variance d'estimation de la DSP d'un bruit blanc évolue-t-elle avec un nombre d'échantillons N croissant ? Accord entre théorie et pratique ?
— Comment la variance d'estimation de la DSP d'un processus AR1 évolue-t-elle avec un nombre d'échantillons N croissant ? Accord entre théorie et pratique ?
— Comment la variance d'estimation de la DSP d'une sinusoïde à phase aléatoire évolue-t-elle avec un nombre d'échantillons N croissant ? Accord entre théorie et pratique ?
— Parvient-on à améliorer la variance d'estimation avec un moyennage ? Si oui, quel est le prix à payer ?

4 Choix d'une fenêtre temporelle

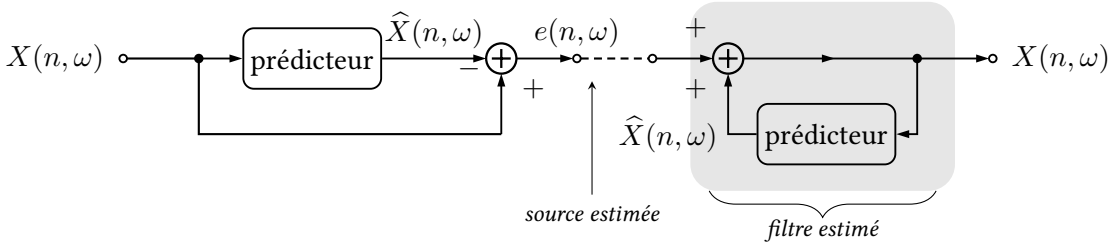
Selon la nature spectrale du signal observé, on peut vouloir pondérer les échantillons dont on dispose à l'aide de différentes fenêtres.

Liste non exhaustive de questions que l'on peut se poser :
— Quel type de tracé est le plus adéquat pour la représentation de la DSP d'un processus présentant des raies ?
— Quelle fenêtre choisir quand on observe le spectre d'un signal très peu structuré ?
— Comment normaliser les coefficients de la fenêtre pour qu'avec un bruit blanc, on puisse retrouver la valeur de la variance dans la DSP estimée ?
— Quelle fenêtre choisir quand on s'intéresse à la fréquence précise du fondamental d'un son voisé ?
— Quelle fenêtre choisir quand on s'intéresse à l'amplitude d'une harmonique composant un son voisé ? Comment normaliser les coefficients de la fenêtre pour que la hauteur du pic correspondant donne directement l'amplitude ?
— Comment la variance d'estimation de la DSP d'un processus AR1 évolue-t-elle avec un nombre d'échantillons N croissant ? Accord entre théorie et pratique ?
— Comment la variance d'estimation de la DSP d'une sinusoïde à phase aléatoire évolue-t-elle avec un nombre d'échantillons N croissant ? Accord entre théorie et pratique ?
— Peut-on établir un critère fiable basé sur l'estimateur de DSP pour distinguer les sons voisés des sons non voisés ? Si oui, que vaut-il par rapport à celui établi sur l'autocorrélation ?

S3 Prédiction linéaire

Objectif de cette troisième séance : mettre en œuvre la prédiction linéaire afin d'établir un modèle source/filtre des sons voisés et non voisés constituant la parole.

La prédiction linéaire optimale permet de décorréler les échantillons du signal. Plus précisément, à un ordre K, l'erreur de prédiction $e(n,\omega)$ est décorrélée des K derniers échantillons du signal $X(n,\omega)$. On espère qu'ainsi, pour des sons vocaux, l'erreur de prédiction puisse fournir une estimation de la *source d'excitation* et que les coefficients optimaux donnent un bon modèle de *filtre*.



1 Résolution des équations de Yule-Walker

Écrire la fonction résolvant le système d'équations de Yule-Walker et décrite de la manière suivante :

```
%% [a,v] = YuleWalkerSolver(Cx,K)
% Solves the Yule-Walker equations for a given cross-correlation
% sequence. This function can be used in two scopes:
% 1) finding optimized linear prediction coefficients at order K, such
%     that: Xest(n+1) = a(1)*X(n) + a(2)*X(n-1) + ... + a(K)*X(n-K+1).
%     Output v represents the minimal variance of prediction error.
% 2) fitting the parameters of an AR(K) process generated by
%     X(n) = a(1)*X(n-1) + a(2)*X(n-2) + ... + a(K)*X(n-K) + W(n),
%     where W is a white noise, called innovation, with variance v.
%
% INPUTS
% - Cx    cross-correlation vector for lags 0 to K
% - K     order ( optional argument, defaults to length(Cx)-1 )
% OUTPUTS
% - a     vector of filter coefficients.
% - v     prediction error variance or innovation power
function [a,v] = YuleWalkerSolver(Cx,K)
```

Conseil : on pourra utiliser la fonction `toeplitz` et l'opérateur `\`.

Grâce aux coefficients fournis par cette fonction, on peut vérifier la capacité de prédiction pour des processus connus.

◇ **Checkpoint** 7: Dans un premier temps, testez votre fonction en utilisant directement les *coefficients d'autocorrélation théorique* d'un processus aléatoire. Le tableau suivant rassemble quelques exemples de processus caractérisés par une équation de récurrence.

Nom du processus	Autocorrélation $\gamma_X[p] =$	Équation de récurrence $X(n,\omega) = \cdots$
AR(1)	$\sigma^2 a^{\lvert p\rvert}/(1-\lvert a\rvert^2)$	$B(n,\omega) + aX(n-1,\omega)$
AR(P)	$\sigma^2\gamma_h[p]$ avec h rép. imp.	$B(n,\omega) + \sum_{k=1}^{P} a_k X(n-k,\omega)$
Sinusoïde à phase aléatoire	$\lvert a\rvert^2 \cos(2\pi\nu_0 p)/2$	$2\cos(2\pi\nu_0)X(n-1,\omega) - X(n-2,\omega)$

Liste non exhaustive de questions que l'on peut se poser :
- Que faire dans la fonction `YuleWalkerSolver` si le vecteur `Cx` fournit contient moins de $K+1$ éléments ?
- Avec l'autocorrélation de quel processus connu peut-on tester la fonction à l'ordre $K = 1$? Quels coefficients optimaux attend-t-on ?
- Avec l'autocorrélation de quel processus connu peut-on tester la fonction à l'ordre $K = 2$? Quels coefficients optimaux attend-t-on ?
- Comment tester le bon fonctionnement de la fonction à un ordre K supérieur ?
- Que se passe-t-il avec un AR(1) et $K > 1$?
- Que se passe-t-il avec une sinusoïde à phase aléatoire et $K > 2$?

2 Réalisations de processus connus

◇ **Checkpoint** 8: Dans un deuxième temps, testez votre fonction en utilisant *des autocorrélations estimées à partir d'un nombre fini d'échantillons de réalisations de processus aléatoires*, par exemple des signaux AR(1), sinusoïdes à phase aléatoires et signaux AR(K).

Conseil : utiliser toujours l'estimateur *biaisé* d'autocorrélation.

Liste non exhaustive de questions que l'on peut se poser :
- Comment évolue la variance d'erreur de prédiction en fonction de l'ordre K ? Pour un processus donné, quel ordre K choisir ?
- Comment la prédiction se compare-t-elle à une réalisation donnée ?
- Comment l'erreur de prédiction se compare-t-elle à une réalisation donnée ?
- Quelle est l'allure de l'autocorrélation de l'erreur de prédiction en fonction du décalage p ? Avec quel processus peut-on modéliser cette erreur ?
- Comment l'autocorrélation d'erreur de prédiction se compare-t-elle à l'autocorrélation du signal ?
- En modélisant le signal par un processus AR(K) défini par les paramètres fournis par la prédiction linéaire, quelle est l'allure de la DSP estimée ? Dans quelle mesure concorde-t-elle avec la DSP théorique ?

3 Sons voisés et non voisés

◇ **Checkpoint** 9: Reprendre l'étude précédente avec les sons voisés et non voisés fournis.

Liste non exhaustive de questions que l'on peut se poser :
- À quel ordre K semble-t-il raisonnable de limiter la prédiction linéaire pour les sons voisés ? Pour les sons non voisés ?
- Selon le caractère voisé du signal, quelle est l'allure de l'autocorrélation de l'erreur de prédiction en fonction du décalage p ? Avec quel processus peut-on modéliser ces erreurs ?
- Comment les estimations spectrales paramétriques se comparent-elles à celles non paramétriques réalisées en séance précédente ?

Finalement, disposant de modèles de filtres et de processus d'excitation, nous sommes en mesure de générer des sons voisés et non voisés synthétiques. À partir des caractéristiques extraites des sons fournis, générer des sons synthétiques sur des durées de plusieurs secondes et vérifier la qualité sonore de cette synthèse.

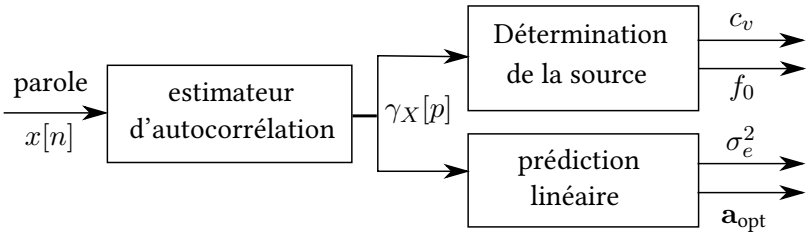
S4 Codage, décodage et effets sur la parole

Objectif de cette dernière séance : mettre en œuvre le codage/décodage du signal de parole par prédiction linéaire en traitant le signal par tronçon. On comparera les signaux réels et synthétiques avec les outils de traitement du signal (autocorrélation, spectre) et à l'oreille. En modifiant les caractéristiques analysées sur les tronçons, on pourra appliquer des effets sur la voix.

Il s'agit maintenant de rassembler des différents outils développés lors des séances précédentes de manière à analyser et synthétiser des signaux vocaux de longue durée. Si vous avez rencontré des difficultés avec l'implémentation de certaines fonctions, il est possible, dans une certaine mesure de s'en affranchir. La table suivante montre les équivalences avec des fonctions de base.

Estimateur	Séances précédentes	Fonction Matlab (*signal processing toolbox*)	Fonction Octave (*signal package*)
Autocorrélation	`[Un]BiasedCrossCorr`	`xcorr`	`xcorr`
Spectre	`psdEstimator`	`pwelch`	`pwelch`
Filtre de prédiction	`YuleWalkerSolver`	`aryule` ou `lpc`	`yulewalker` ou `lpc`

1 Analyse d'un tronçon



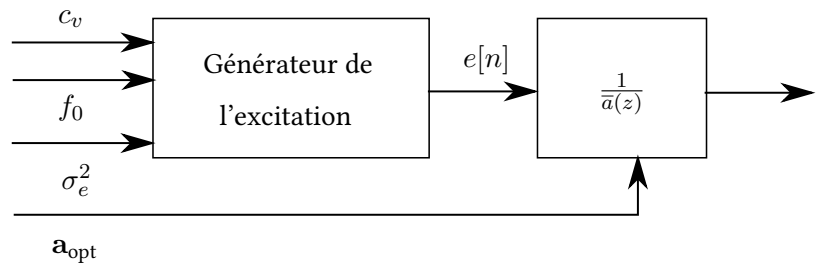
Écrire la fonction permettant d'analyser un tronçon de signal de 256 échantillons, décrite de la manière suivante :

```
%% [pitch,sigma2,Aopt] = BlockAnalysis(X,M,Fe)
% Analyses the samples of a supposedly stationnary process
% and fits the parameters of a model source/filter.
%
% INPUTS
% - X      vector of samples of the current block of signal
% - M      filter order (optional, defaults to 10)
% - Fe     sampling frequency (optional, defaults to 10kHz)
% OUTPUTS
% - pitch  the pitch value in normalized frequency if the sound
%          is voiced, -1 otherwise.
% - sigma2 the variance of excitation
% - Aopt   vector of filter coefficients (optimal linear prediction
%          of order M).
function [pitch,sigma2,Aopt] = BlockAnalysis(X,M,Fe)
```

On pourra s'appuyer sur une sous-fonction répondant à la définition suivante :

```
%% pitch = PitchDetector(Cx,Fe,Fmin,Fmax)
% Determines the pitch of a signal based on its crosscorrelation.
%
% INPUTS
% - Cx     vector of cross-correlation samples for shifts varying
%          from 0 to pmax
% - Fe     sampling frequency (optional, defaults to 10kHz)
% - Fmin   minimal pitch (optional, defaults to 100Hz)
% - Fmax   maximal pitch (optional, defaults to 400Hz)
% OUTPUTS
% - pitch  the pitch value in normalized frequency if the sound
%          is voiced, -1 otherwise.
function pitch = PitchDetector(Cx,Fe,Fmin,Fmax)
```

2 Synthèse d'un tronçon



Écrire la fonction permettant de synthétiser un tronçon de signal, décrite de la manière suivante :

```
%% Y = BlockSynthesis(pitch,sigma2,Aopt,N)
% Synthetizes a random signal based on the parameters of a
% source/filter model.
%
% INPUTS
% - pitch  the pitch value in normalized frequency if the sound
%          is voiced, -1 otherwise.
% - sigma2 the variance of excitation
% - Aopt   vector of optimal filter coefficients.
% - N      number of samples to be synthetized
% OUTPUTS
% - Y      synthetized block
function Y = BlockSynthesis(pitch,sigma2,Aopt,N)
```

3 Traitement d'un signal et effets

Commencez par écrire un script chargeant un signal sonore, l'analysant par tronçons puis synthétisant de nouveaux tronçons à partir des caractéristiques extraites.

◇ **Checkpoint** 10: Testez d'abord votre script pour un seul tronçon, en comparant visuellement la reconstruction avec le signal de départ (en termes d'autocorrélation ou de DSP).

◇ **Checkpoint** 11: Assemblez les tronçons synthétisés et apprécier à l'écoute la qualité du signal produit.

Conseil : Si les tronçons sont contigus sans recouvrement, les variations de puissance rapides en passant d'un tronçon à l'autre perturbent la qualité d'écoute. Pour pallier ce problème, on peut utiliser des tronçons pondérés par une fenêtre de Hann avec un recouvrement de 50%.

Effets que l'on peut envisager

- Changement de tonalité sans changement de durée
- Passage à une voix chuchotée (toujours du bruit blanc pour l'excitation)
- Voix de robot (toujours une impulsion périodique pour l'excitation avec un fondamental fixe, indépendamment du signal analysé)
- Remplacement de l'excitation par une piste instrumentale.