TRAVAUX PRATIQUES DE TRAITEMENT DU SIGNAL

Volume 1

Marie Chabert et Corinne Mailhes

Année d'Edition : 2010

Table des matières

Avant-Propos

Un ensemble de logiciels de Travaux Pratiques pour le Traitement du Signal a été élaboré par les chercheurs de l'équipe Signal et Communication (SC) de l'Institut de Recherche en Informatique de Toulouse (IRIT), également membres du TéSA (Laboratoire de Télécommunications Spatiales et Aéronautiques). Les domaines de recherche abordés par ce groupe de chercheurs sont variés : analyse spectrale, classification et reconnaissance des formes, compression de données, traitement statistique du signal, télécommunications spatiales... Pour plus de renseignements, on peut se reporter aux sites Web :

$$\begin{aligned} http &: //www.irit.fr/-Equipe-SC- \\ http &: //www.tesa.prd.fr \end{aligned}$$

Le Traitement du Signal ne doit pas être vu seulement de façon théorique, au niveau des cours et des Travaux Dirigés ; le Traitement du Signal est une matière vivante, actuellement en plein essor, avec des applications de plus en plus nombreuses, dans des domaines très variés. Donnons deux exemples d'études effectuées au SIC dans le cadre de doctorats.

Une première étude concerne le système d'aide à l'atterrissage des avions (ILS : Instrument Landing System). Ce système, adopté internationalement par l'Aviation Civile, utilise pour sa partie radioalignement de piste une bande de fréquences VHF qui est, depuis quelques années, très vulnérable à des brouilleurs externes, comme les radios FM, les brouilleurs industriels etc... Il est très important d'avoir une connaissance sur ces brouilleurs, aussi le Service Technique de la Navigation Aérienne a-t-il suscité une étude qui a fait l'objet d'une thèse. Le problème a été abordé à l'aide des méthodes de Traitement du Signal. Le signal qu'on désire conserver est un signal de faible puissance (le brouilleur) par rapport au signal ILS utilisé à l'atterrissage qui doit être estimé, puis rejeté le plus parfaitement possible afin de ne pas nuire à l'identification du brouilleur. Ce travail a mis en jeu des méthodes d'*estimation et de réjection* des deux sinusoïdes qui constituent le signal ILS, puis des méthodes *de segmentation et de classification* permettant de reconnaitre les brouilleurs.

Une deuxième application relève du domaine biomédical. L'usage de l'électromyographie (EMG) en tant qu'indicateur de l'état du système neuromusculaire remonte à la fin du XIXème

siècle. Cette technique est aujourd'hui très importante dans la détection et le suivi d'atteintes nerveuses ou musculaires. Il s'agit d'observer l'activité électrique émise lors d'une contraction naturelle du muscle. Pendant longtemps, ces signaux étaient recueillis au moyen d'une électrode qu'on *piquait* dans la masse musculaire du patient. Outre l'aspect traumatisant de la méthode, le diagnostic s'avérait parfois difficile. D'où l'idée d'utiliser des électrodes de surface, nullement effrayantes pour le malade. Malheureusement, l'inspection visuelle du signal recueilli n'est pas suffisante pour transmettre la caractérisation d'une pathologie neuromusculaire. Les méthodes de Traitement du Signal, en particulier les méthodes d'analyse spectrale, de modélisation paramétrique et de classification, ont conduit à l'élaboration d'un outil d'*aide au diagnostic* pour des signaux EMG recueillis en surface.

Pour mettre en évidence cet aspect *application pratique* du Traitement du Signal, les chercheurs du SIC ont décidé de créer un ensemble de Travaux Pratiques, venant compléter l'enseignement du Traitement du Signal de l'ENSEEIHT.

En deuxième année, sont prévus quatre TP illustrant les cours de **Théorie du Signal (TS)** et de **Traitement Numérique du Signal (TNS).** Le premier TP TS s'intéresse aux **probabilités élémentaires**, le deuxième traite de l'**échantillonnage et de la quantification.** Le premier TP TNS s'intéresse au calcul des fonctions de **corrélations** et des **densités spectrales de puissance** et le deuxième au **filtrage numérique**.

En troisième année, les TP sont plus spécifiques : deux TP sur l'estimation paramétrique et non paramétrique (cours de **Représentation et Analyse des Signaux** et de **Modélisation Paramétrique**) et un TP sur le filtrage adaptatif (cours de **Traitement Adaptatif**).

Ces TP ont été organisés autour de logiciels MATLAB. Ces logiciels ont été conçus pour être utilisés à partir de menus déroulants, évitant à l'utilisateur de se plonger dans la lourde tâche de la programmation.

Des indications de programmation Matlab, données en fin de chaque TP, associées à une initiation Matlab donnée en annexe, permettent de reprogrammer facilement certaines fonctionnalités des logiciels. Ceci peut constituer une aide à la réalisation des projets Matlab de deuxième et troisième année.

Chapitre 1

TP Probabilités et Statistiques

Lancer `prob_stat` sous Matlab.

1.1 Etude de l'histogramme d'amplitude

1.1.1 Variables aléatoires discrètes - Estimation de la probabilité

Comment obtenir un estimateur de la loi de probabilité d'une variable aléatoire discrète à partir de l'histogramme de N réalisations de cette variable? En déduire un estimateur de la fonction de répartition.

A l'aide du logiciel, observer les performances de ces estimateurs en fonction du nombre de réalisations N pour les variables aléatoires :

• uniforme sur $\{1, ..., P1\}$

• binomiale $B(P1 = n, P2 = p)$

• de Poisson de paramètre $\lambda = P1$.

Le bouton `OK` permet de générer N nouvelles réalisations de la variable aléatoire. Noter les moments estimés en fonction de N (moyenne et variance) et comparer à la théorie.

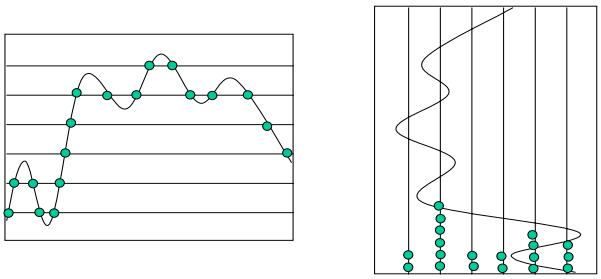
1.1.2 Variables aléatoires continues - Estimation de la densité de probabilité

La probabilité pour que la variable (v.a.) X soit comprise entre x et $x + dx$ peut être approchée, pour dx petit, par $p(x)dx$, $p(x)$ étant la densité de probabilité de la v.a. X. La probabilité qu'une

réalisation de la v.a. X soit comprise entre mq et $(m+1)q$ est :

$$P[m] = \int_{mq}^{(m+1)q} p(x)dx$$

Considérons une fonction représentée graphiquement sur un boulier. Sur ce boulier sont placées des billes qui représentent plus ou moins finement la fonction : les tiges peuvent être assez rapprochées et les billes assez petites pour donner une impression de continuité selon les deux axes de coordonnées.



Si on place ce boulier dans le plan vertical et qu'ensuite, on lui fait subir une rotation de 90° dans ce plan vertical, les billes vont glisser le long des fils et on aura le nombre de billes sur chaque fil, image (grossière bien sûr) de la densité de probabilité. C'est exactement l'opération que l'on réalise lorsqu'on fait l'histogramme d'amplitude. C'est ce que nous nous proposons de faire dans ce TP : utiliser l'histogramme pour estimer la densité de probabilité d'une variable aléatoire.

Soit $Z(t)$ la variable indicatrice définie par :

$$\begin{aligned} Z(t) &= 1 \text{ si } X(t) \in \left[x - \frac{\Delta x}{2}, x + \frac{\Delta x}{2}\right] \\ Z(t) &= 0 \text{ si } X(t) \notin \left[x - \frac{\Delta x}{2}, x + \frac{\Delta x}{2}\right] \end{aligned}$$

Calculons la moyenne de cette variable :

$$\begin{aligned} E\left\{Z\left(t\right)\right\} &= 1.P\left[Z\left(t\right) = 1\right] + 0.P\left[Z\left(t\right) = 0\right] \\ &= P\left[Z\left(t\right) = 1\right] = P\left[X(t) \in \left[x - \frac{\Delta x}{2}, x + \frac{\Delta x}{2}\right]\right] \end{aligned}$$

On obtient finalement :

$$E\left\{Z\left(t\right)\right\} \cong p\left(x\right)\Delta x \text{ si } \Delta x \text{ suffisamment petit}$$

Estimer la densité de probabilité revient à construire un estimateur de moyenne de la variable indicatrice. On définit des classes, c'est-à-dire que l'axe des y est divisé en intervalles de même

largeur Δx. On note $Z_i(t)$ la variable définie de la même façon que $Z(t)$ sur l'intervalle n°i. L'estimateur habituel de la moyenne de cette v.a. $Z_i(t)$ est :

$$\frac{1}{N}\sum_{t=1}^{N} Z_i(t) = \frac{N_x}{N}$$

N_x étant le nombre d'observations de $X(t)$ appartenant à l'intervalle considéré de largeur Δx et N étant le nombre total d'observations de $X(t)$. D'où l'estimation de la densité de probabilité suivante :

$$\widehat{p}(x) \approx \frac{N_x}{N\Delta x}$$

où N_x "Nombre d'échantillons $\in [x - \frac{\Delta x}{2}, x + \frac{\Delta x}{2}]$" représente l'histogramme. Les valeurs minimale et maximale prises par la variable aléatoire sur les réalisations considérées et le nombre d'intervalles (ou nombre de classes) permettent de déterminer la valeur de Δx.

Un estimateur est caractérisé par son biais et sa variance.

Soit $\widehat{\theta}$ un estimateur d'un paramètre θ. Le biais de cet estimateur est défini par :

$$biais(\theta) = E\left\{\widehat{\theta}\right\} - \theta$$

En d'autres termes, le biais représente l'erreur systématique que l'on commet lorsqu'on estime θ. La variance de $\widehat{\theta}$ permet de mesurer la vitesse de convergence de $\widehat{\theta}$ vers θ lorsque N augmente. Un estimateur asymptotiquement sans biais

$$\lim_{N\to\infty} E\left\{\widehat{\theta}\right\} = \theta$$

et dont la variance vérifie :

$$\lim_{N\to\infty} var\left\{\widehat{\theta}\right\} = 0$$

est convergent. Etudions les propriétés de l'estimateur de la densité de probabilité que nous avons défini, $\widehat{p}(x)$. Pour cela, calculons la moyenne de la variable indicatrice $Z(t)$:

$$E\{Z(t)\} = \int_{x-\frac{\Delta x}{2}}^{x+\frac{\Delta x}{2}} p(u)du$$

Pour Δx, largeur d'une classe de l'histogramme suffisamment petite et $p(x)$ régulière, par un développement en série de Taylor, on a :

$$E\{Z(t)\} \approx \left[p(x) + \frac{\Delta x^2}{24}p"(x)\right]\Delta x$$

L'estimateur de densité de probabilité possède donc un biais additif :

$$b\{\widehat{p}(x)\} \approx \frac{\Delta x^2}{24}p"(x)$$

Si les échantillons sont indépendants, alors la variance est égale à :

$$Var\{\widehat{p}(x)\} \approx \frac{1}{N\Delta x^2} var[Z(t)] \approx \frac{p(x)}{N\Delta x}$$

On retrouve le dilemme biais-variance où, pour N fixé, la variance et le biais évoluent en sens inverse de Δx.

1.1.3 *Estimation de la fonction de répartition*

La fonction de répartition F se définit de la façon suivante :

$$F(x) = P[X < x] = \int_{-\infty}^{x} p(u)du$$

Pour estimer la fonction de répartition F, on considère une autre variable indicatrice $Z(t)$ telle que :

$$\begin{aligned} Z(t) &= 1 \text{ si } X(t) \leq x \\ Z(t) &= 0 \text{ si } X(t) > x \end{aligned}$$

Alors, on a :

$$E\{Z(t)\} = F(x)$$

L'estimateur de la fonction de répartition est un estimateur de moyenne dans le cas d'échantillons indépendants. Sa variance est telle que :

$$var\left\{\widehat{F}(x)\right\} \approx \frac{1}{N} F(x)(1 - F(x))$$

Cet estimateur est non biaisé et possède une variance qui est minimale pour :

$$F(x) \approx 0 \text{ et } F(x) \approx 1$$

Remarquons que les estimateurs de la densité de probabilité et de la fonction de répartition ont tous les deux une variance en $1/N$.

1.1.4 Travail à effectuer

Estimation de la densité de probabilité

Générer $N = 1000$ réalisations d'une variable aléatoire gaussienne centrée réduite. En faire l'histogramme sur $N_c = 300$ classes.

Comment obtient-on l'estimateur de densité de probabilité à partir de l'histogramme? Observer l'estimation de sa densité de probabilité. Etudier l'influence du nombre de classes, à nombre d'échantillons constant, sur la variance de l'estimateur (en utilisant la fonction "superposition des tracés").

Observer la densité de probabilité estimée d'une variable aléatoire uniformément répartie. Noter les valeurs des moments estimés et comparer à la théorie.

Le programme "`moy_e_dp`", indépendant du logiciel, permet d'étudier les propriétés de l'estimateur de densité de probabilité en termes de biais et de variance en fonction du nombre de classes de l'histogramme et du nombre d'échantillons. La moyenne de l'estimateur ainsi que sa variance

sont estimées sur un nombre de réalisations qu'il est également possible de modifier. Le programme trace la moyenne (en vert), la moyenne plus ou moins l'écart type (en magenta) ainsi que la densité de probabilité théorique (en rouge). Observer les propriétés de l'estimateur en fonction du nombre de classes N_c et du nombre de réalisations de la variable aléatoires N.

Estimation de la fonction de répartition

Déterminer la fonction de répartition de la gaussienne. Observer le biais et la variance de cet estimateur (utiliser la fonction "`superposition des tracés`" pour superposer plusieurs réalisations.

Observer la fonction de répartition d'une variable aléatoire uniformément répartie sur $[0, 1]$.

1.2 Loi des grands nombres

Le traitement "grands nombres" permet de tracer en fonction de N l'estimateur de la moyenne de X calculé sur N échantillons :

$$\widehat{M}_N = \frac{1}{N}\sum_{k=1}^{N} x(k)$$

Observer le comportement de cet estimateur en fonction de N pour des variables aléatoires générées suivant les distributions suivantes :

- loi normale centrée réduite,
- loi uniformément répartie,
- loi exponentielle
- loi de Cauchy.

1.3 Théorème Central-Limite - Test de Kolmogorov

1.3.1 *Théorème central-limite*

Soient X_k, $k = 1, \cdots, K$, des variables aléatoires indépendantes et de même loi, de moyenne m et de variance σ^2. On pose

$$S_K = \frac{1}{K}\sum_{k=1}^{K} X_k$$

alors

$$\lim_{K\to+\infty} \frac{S_K - m}{\frac{\sigma}{\sqrt{K}}} \overset{\text{en loi}}{=} N(0,1)$$

c'est-à-dire que l'on sait construire une loi normale centrée réduite à partir de variables aléatoires de n'importe quelle loi indépendantes et de même loi. Ce théorème est très puissant et

très important. Il va permettre de traiter de nombreux problèmes en faisant l'hypothèse de loi gaussienne pour le phénomène observé.

1.3.2 *Test de Kolmogorov*

Le test de Kolmogorov permet de décider (avec une marge d'erreur donnée) si une variable aléatoire X suit une loi de fonction de répartition $F(x)$. Le déroulement du test est le suivant :

1) Estimation de la fonction de répartition $\widehat{F}(x)$ à partir de K observations x_k $k = 1, ..., K$ de la variable aléatoire X.

2) Recherche de l'écart maximum $\Delta_{\max}$ entre $\widehat{F}(x)$ et $F(x)$.

3) Pour un risque de première espèce α donné (1% ou 5% généralement) vérifier que la valeur donnée dans la table de Kolmogorov est supérieure à $\Delta_{\max}$. Si c'est le cas, on décide que X suit la loi de fonction de répartition $F(x)$ avec un risque d'erreur α%.

Rappelons la notion de risque lorsqu'on fait un test d'hypothèse H_0 contre H_1 : 2 risques sont définis :

$\alpha = P$[rejeter H_0 sachant H_0 vraie] *risque de non détection*

$\beta = P$[accepter H_0 sachant H_0 fausse] *risque de fausse alarme*

Le test de Kolmogorov, très simple à mettre en oeuvre, ne prend malheureusement en compte que le risque de première espèce α.

1.3.3 Travail à effectuer

A l'aide de la fonction "`Th. Central-Limite`" et du bouton "`OK`", il est possible de générer la somme de K variables aléatoires uniformément réparties sur l'intervalle $[0, 1]$, X_1 ... X_K. Soit S_K la variable telle que :

$$S_K = \frac{1}{K} \sum_{j=1}^{K} X_j$$

Générer $N = 500$ réalisations et choisir un nombre de classes pour l'histogramme $N_c = 500$. A l'aide du théorème central-limite, déterminer la loi de S_K quand K tend vers l'infini. Observer la moyenne et la variance ainsi que la densité de probabilité. Expliquer.

A l'aide du test de Kolmogorov, étudier à partir de quelle valeur de N on peut considérer avoir une loi de Gauss. Le logiciel effectue le test suivant les étapes suivantes :

1 - Fonction "`Théorème Central-Limite`" du logiciel :

Génération et sommation de K séquences aléatoires (de N échantillons) indépendantes de même loi, en l'occurence uniformément réparties sur $[0, 1]$.

Estimation de la fonction de répartition (sur Nc classes) de la somme S_K de ces K séquences aléatoires.

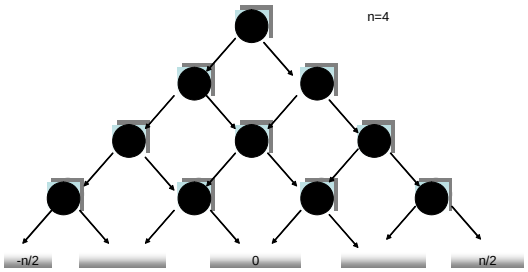
Estimation de la moyenne $\widehat{m}$ et de la variance $\widehat{\sigma}^2$ de Y_N.

2 - Fonction "`Kolmogorov normal?`"
Calcul théorique par le logiciel de la fonction de répartition de la gaussienne ayant pour moyenne $\widehat{m}$ et pour écart type $\widehat{\sigma}$.
Calcul par le logiciel de l'écart maximum entre cette fonction de répartition théorique et la fonction de répartition estimée de Y_N.
Comparaison avec l'écart maximal donné dans les tables.

1.4 Etude de distributions particulières

1.4.1 Triangle de Galton

Il s'agit d'un plan incliné hérissé de clous régulièrement disposés sur n lignes horizontales, la $k^{ième}$ ligne comportant k clous. Une bille, placée au sommet du triangle, oblique, à chaque ligne, vers la droite ou la gauche avec une probabilité $\frac{1}{2}$. Les billes s'amassent dans les $n+1$ cases numérotées de $-\frac{n}{2}$ à $\frac{n}{2}$qui constituent la dernière ligne.



Triangle de Galton pour n=4

Pour observer la trajectoire d'une bille, sélectionner dans le menu signal "`bille de Galton`". Le nombre de lignes est réglable (pour cela modifier le "nombre de réalisations"). On prendra $n = 50$.

On note X la variable aléatoire correspondant au point de chute de la bille (indice de la case de la dernière ligne où la bille est tombée). La bille aboutit dans la $k^{ième}$ case en partant de la gauche si elle a obliqué k fois vers la droite et $n-k$ fois vers la gauche. On note X_i la variable aléatoire asssociée à chaque rebond

$$X_i = 0 \text{ la bille oblique à gauche}$$
$$X_i = 1 \text{ la bille oblique à droite}$$

Vérifier que $X = \sum_{j=1}^{n} X_j - \frac{n}{2}$. Quelle est la loi de $X + \frac{n}{2}$?

Grâce au menu "`demos`" $\longrightarrow$ "`Galton pas à pas`" il est possible de lancer successivement l billes (cliquer sur `OK`) et d'observer le nombre de billes tombées dans chacune des cases.

En déduire une approximation de la loi de X pour l grand. Vérifier l'hypothèse grâce au traitement "`Galton n grand`" (il permet de faire tomber l billes sans avoir à cliquer pour chacune d'elle).

Chi-Deux

Soient X_k, $k = 1, \cdots, K$, des variables aléatoires indépendantes et de loi $N(0,1)$. La variable v_K

$$V_K = \sum_{k=1}^{K} X_k^2$$

suit une loi du Chi-Deux à K degrés de liberté. La moyenne et la variance de cette variable aléatoire sont :

$$E[V_K] = K$$

$$var[V_K] = 2K$$

Générer des signaux de lois Chi-Deux de degrés croissant (bouton "`OK`"). Comparer les moyennes et les variances estimées aux valeurs théoriques. Que se passe-t-il lorsque le degré du Chi-Deux devient très grand. A partir de quel degré le test de Kolmogorov accepte l'hypothèse gaussienne?

Box-Muller

Dans le menu distribution, le choix "Box-Muller" permet de générer à partir de deux variables aléatoires indépendantes de mêmes lois uniformes sur $]0,1]$ U et V la variable aléatoire X de la façon suivante :

$$X = \sqrt{-2\ln U}\cos(2\pi V)$$

Quelle est la loi de X ?

1.5 Test de Neyman-Pearson

Soient $X_1, ..., X_n$ n variables aléatoires indépendantes de loi $N(m, \sigma^2)$ avec σ^2 connue. On souhaite réaliser le test d'hypothèses :

$$\begin{aligned} H_0 &: m = m_0 \\ H_1 &: m = m_1 > m_0 \end{aligned}$$

D'après le lemme de Neyman-Pearson, la région de rejet de H_0 est donnée par

$$\frac{f(X_1, ..., X_n|H_1)}{f(X_1, ..., X_n|H_0)} > s(\alpha)$$

où $s(\alpha)$ est le seuil de décision, fonction de la probabilité de fausse alarme α. Ce test permet de maximiser la puissance du test $\pi = 1 - \beta$. Dans ce cas, on obtient :

$$\sum_{i=1}^{n} X_i > \lambda(\alpha)$$

Expliquer intuitivement ce résultat.

Le programme `puissance_th(m0,m1,sigma2,n)` renvoie la puissance théorique du test pour n variables pour les valeurs $\alpha = 0.01, 0.02, ..., 0.99$.

Tester ce programme pour $m_0 = 0$, $m_1 = 0.5$, $\sigma^2 = 1$ et $n = 10$. Recommencer avec :

- $m_0 = 0$, $m_1 = 1$, $\sigma^2 = 1$ et $n = 10$;
- $m_0 = 0$, $m_1 = 0.5$, $\sigma^2 = 2$ et $n = 10$;
- $m_0 = 0$, $m_1 = 0.5$, $\sigma^2 = 0.5$ et $n = 10$;
- $m_0 = 0$, $m_1 = 0.5$, $\sigma^2 = 1$ et $n = 20$;
- $m_0 = 0$, $m_1 = 0.5$, $\sigma^2 = 1$ et $n = 50$;

Commenter ces résultats (utiliser la commande `hold on` pour superposer les courbes).

Le programme `puissance_est(m0,m1,sigma2,n,K)` renvoie la puissance du test pour les valeurs $\alpha = 0.01, 0.02, ..., 0.99$ estimée à l'aide de K simulations. Reprendre les questions précédentes avec $K = 50$, puis $K = 500$. Commentaires ?

1.6 Annexes

1.6.1 Demos de la boîte à outils "stats" de Matlab

La boîte à outils Matlab dispose de programmes de démonstration permettant de générer des valeurs aléatoires suivant une loi donnée et d'observer leurs distributions et fonctions de répartition théoriques ou estimées.

Taper `disttool` ou `randtool` sous la fenêtre de commande Matlab.

Pour connaître toutes les fonctions disponibles dans la boîte à outils statistiques taper "`help stats''`.

1.6.2 Rappels de quelques distributions

• Distribution binômiale $\mathcal{B}(n, p)$:

$$P[X = k] = C_n^k p^k q^{n-k} \qquad (k = 0, 1, ..., n; 0 < p < 1; q = 1 - p)$$

• Distribution uniforme sur $[a, b]$: $f(x) = \frac{1}{b-a}$ *pour* $x \in [a, b]$ (0 ailleurs)

$$E(X) = \frac{b+a}{2} \qquad Var(X) = \frac{(b-a)^2}{12}$$

• Distribution Normale $N(m, \sigma^2)$:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left[-\frac{(x-m)^2}{2\sigma^2}\right]$$

• Distribution du Chi-deux à n degrés de liberté :

$$\begin{aligned} f(x) &= \frac{1}{2^{\frac{n}{2}}\Gamma(\frac{n}{2})} x^{\frac{n}{2}-1} e^{-\frac{x}{2}} \quad pour\ x \in [0, \infty] \text{ (0 ailleurs)} \\ E(X) &= n \qquad Var(X) = 2n \end{aligned}$$

1.6.3 Table du Test de Kolmogorov

La table suivante donne l'écart maximal théorique, Δ_{kolmo}, entre fonctions de répartition empirique et théorique pour accepter l'hypothèse H_0 avec un risque α de 5 ou 1%, en fonction de Nc (nombre de points de calcul de la fonction de répartition, c'est-à-dire nombre de classes de l'histogramme). Si l'écart maximal mesuré $\Delta_{\max}$ est inférieur à Δ_{kolmo}, on accepte l'hypothèse H_0 avec un risque α.

N_c	$\alpha = 5\%$	$\alpha = 1\%$
5	0.5633	0.6685
10	0.4087	0.4864
15	0.3375	0.4042
20	0.2939	0.3524
25	0.2639	0.3165
30	0.2417	0.2898
40	0.2101	0.2521
50	0.1884	0.2260
60	0.1723	0.2067
70	0.1597	0.1917
80	0.1496	0.1795
90	0.1412	
100	0.1340	

Cette table n'est valable que pour N_c inférieur à 100. Pour N_c plus grand, on sait calculer $P(\max|\widehat{F} - F| > \Delta_{kolmo})$ sous H_0 (Vladimir N.Vapnik, "The Nature of Statistical Learning

Theory", Springer Ed. p87). Or :

$$\alpha = P\,[\text{rejeter } H_0 \mid H_0 \text{ vraie}] = P(\max|\widehat{F}-F| > \Delta_{kolmo}) \simeq \sum_{k=1}^{k=+\infty} 2(-1)^{k-1}\exp(-2k^2\,N_c\,\Delta^2_{komo})$$

On mesure tout d'abord $\Delta_{\max} = \max|\widehat{F} - F|$, écart maximal entre la fonction de répartition théorique et la fonction de répartition estimée. On calcule ensuite :

$$P_{\Delta_{\max}} = P(\max|\widehat{F} - F| > \Delta_{\max}) \simeq \sum_{k=1}^{k=+\infty} 2(-1)^{k-1}\exp(-2k^2\,N_c\,\Delta^2_{\max})$$

Or cette probabilité est une fonction décroissante de $\Delta_{\max}$. Donc si $P_{\Delta_{\max}} > \alpha$, on peut en déduire que $\Delta_{\max} < \Delta_{kolmo}$. Il n'est donc pas nécessaire de calculer Δ_{kolmo} pour prendre la décision. En effet, on accepte l'hypothèse H_0 si :

$$\sum_{k=1}^{k=+\infty} 2(-1)^{k-1}\exp(-2k^2\,N_c\,\Delta^2_{\max}) > \alpha$$

Le calcul de cette probabilité est effectuée par le logiciel.

Chapitre 2

TP Echantillonnage et Quantification

2.1 Rappels

L'échantillonnage et la quantification permettent l'acquisition et le codage d'un signal en vue d'un traitement ou d'un stockage. Le but de ce TP est d'illustrer les notions d'échantillonnage et de quantification et d'analyser leurs effets sur différents signaux.

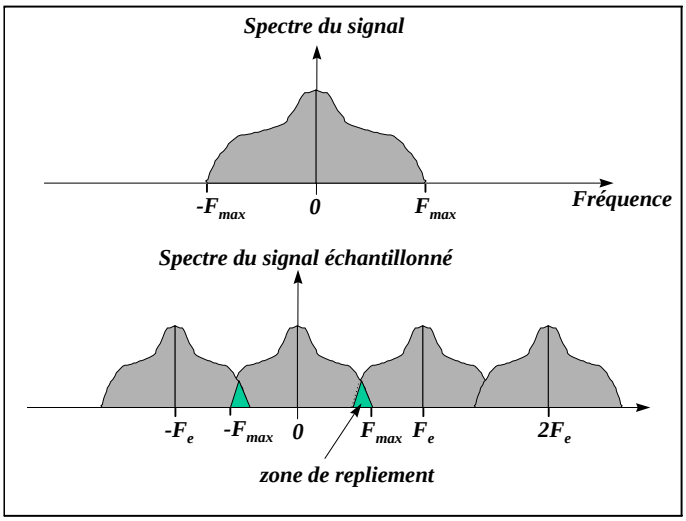
2.1.1 Echantillonnage

L'échantillonnage consiste à représenter un signal à temps continu $s(t)$ par ses valeurs $s(nT_e)$ à des instant multiples de T_e, T_e étant la **période d'échantillonnage**. La **condition de Shannon** permet d'échantillonner un signal sans perte d'information aucune perte d'information si la fréquence d'échantillonnage $f_e = \frac{1}{T_e}$ est au moins 2 fois supérieure à la plus grande fréquence intervenant dans le spectre (répartition de la puissance du signal en fonction des fréquences) du signal. Si cette condition n'est pas respectée on observe un "repliement de spectre" (voir figure suivante). On note ce signal échantillonné $s_e(t)$:

$$s_e(t) = s(t) \times \sum_{-\infty}^{+\infty} \delta(t - nT_e)$$

Dans le plan fréquentiel :

$$S_e(f) = S(f) * \frac{1}{T_e} \sum_{-\infty}^{+\infty} \delta\left(f - \frac{n}{T_e}\right) = \frac{1}{T_e} \sum_{-\infty}^{+\infty} S\left(f - \frac{n}{T_e}\right)$$



Effet de repliement de spectre

L'échantillonnage à la période T_e a donc introduit une périodicité du spectre du signal échantillonné, de période F_e. Lorsqu'on observe des spectres de signaux échantillonnés, on préfère remplacer les fréquences (F) en Hertz par des fréquences normalisées par rapport à la fréquence d'échantillonnage (F_e) :

$$\widetilde{f} = F/F_e$$

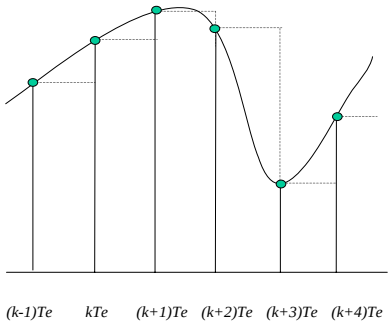
Remarquons qu'après échantillonnage, le spectre entier du signal est disponible entre 0 et 1 en fréquence normalisée. En pratique, par raison de symétrie, on ne s'intéresse qu'à la partie centrale entre 0 et 0.5 en fréquence normalisée.

2.1.2 Restitution aprés échantillonnage du signal d'origine

Plusieurs méthodes d'interpolation existent pour restituer le signal entre deux points d'échantillonnage successifs. Entre deux points d'échantillonnage situés entre kT_e et $(k+1)\,T_e$, comment reconstituer le signal $\widehat{s}\,(kT_e + \tau)$?

Par bloqueur :

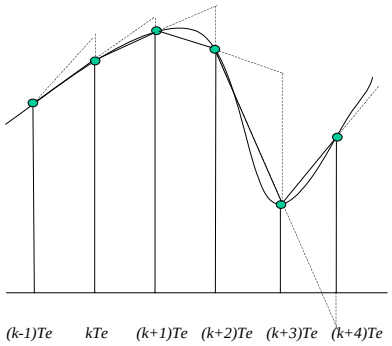
$$\widehat{x}(kT_e + \tau) = x(kT_e) \qquad 0 \leq \tau \leq T_e$$



Par extrapolateur linéaire :

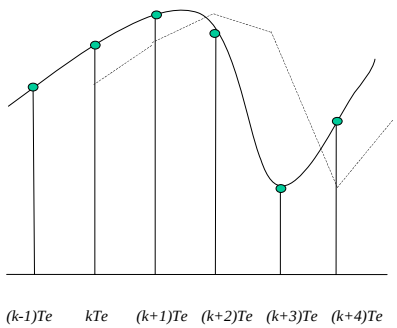
$$\widehat{x}(kT_e + \tau) = x(kT_e) + \frac{\tau}{T_e}\left(x(kT_e) - x((k-1)T_e)\right) \qquad 0 \leq \tau \leq T_e$$

Les marches d'escalier sont remplacées par des rampes.



Par interpolateur linéaire :

$$\widehat{x}(kT_e + \tau) = x(kT_e) + \frac{\tau}{T_e}\left(x((k+1)T_e) - x(kT_e)\right) \qquad 0 \leq \tau \leq T_e$$



Dans la restitution, il y aura un retard de T_e : il est nécessaire d'attendre la valeur en kT_e pour restituer l'intervalle $[(k-1)T_e, kT_e]$.

Par filtrage

Après échantillonnage, le spectre du signal est :

$$S_e(f) = F_e \sum_{n \in Z} S(f - nF_e), \quad F_e = \frac{1}{T_e}$$

Si $x(t)$ est à spectre borné $(-F_M, +F_M)$, et si la condition de Shannon est vérifiée, on peut restituer le signal par filtrage de façon à ne récupérer que le spectre d'ordre 0 :

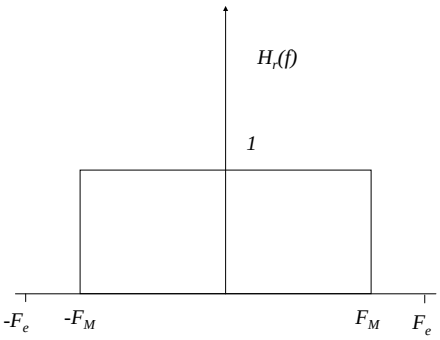
$$S(f) = \frac{1}{F_e} [S_e(f)]_{n=0}$$

Donc

$$S(f) = S_e(f) H_r(f)$$

où $H_r(f)$ est le filtre de restitution :

$$\begin{aligned} H_r(f) &= \frac{1}{F_e} \text{ si } f \in (-F_M, F_M) \\ H_r(f) &= 0 \text{ si } |f| \geq F_e - F_M \end{aligned}$$



Dans le cas limite où $F_e = 2F_M$:

$$H_r(f) = \frac{1}{F_e} \Pi_{F_e}(f)$$

La réponse impulsionnelle associée s'écrit :

$$h_r(t) = \frac{\sin(\pi F_e t)}{\pi F_e t}$$

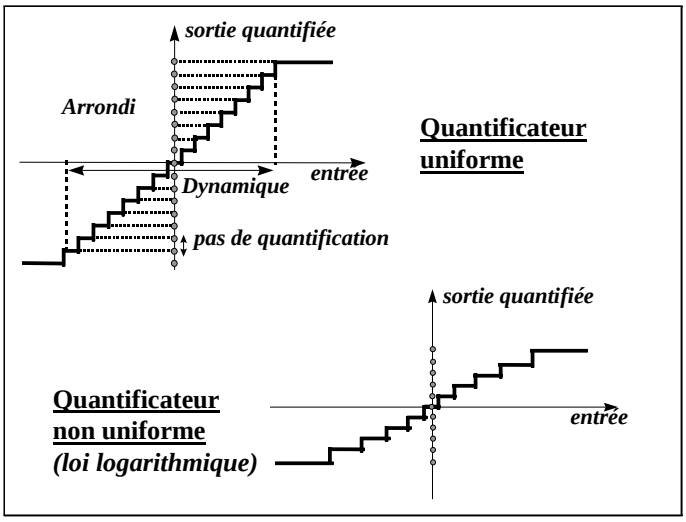
soit, pour le signal restitué :

$$s(t) = \sum_k s_e(kT_e) \frac{\sin\left(\frac{\pi}{T_e}(t - kT_e)\right)}{\frac{\pi}{T_e}(t - kT_e)}$$

La courbe restituée passe exactement par les valeurs des échantillons.

2.1.3 Quantification

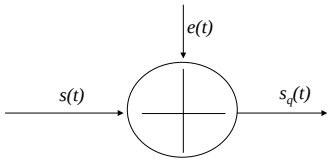
La quantification consiste à allouer aux échantillons un nombre fini de valeurs d'amplitude. Deux grandeurs caractérisent un quantificateur : le nombre de niveaux et la dynamique de codage. On distingue des quantificateurs uniformes (l'écart entre chaque valeur quantifiée est constant) et non uniformes (l'écart entre chaque valeur quantifiée est variable) comme l'illustre la figure suivante.



Quantificateurs uniforme et non uniforme

On note N le nombre de valeurs quantifiées ou nombre de niveaux du quantificateur. Le codage nécessite alors b bits, où 2^b est la puissance de 2 immédiatement supérieure ou égale à N.

L'effet de la quantification revient, en première approximation sous certaines conditions réalisées en pratique (approximation dite de Sheppard) à ajouter au signal $s\,(t)$ un signal d'erreur $e\,(t)$ appelé bruit de quantification, non corrélé avec $s(t)$. Le pas de quantification doit être choisi de façon à minimiser l'erreur en sortie du quantificateur.



Bruit de Quantification

Quantification uniforme

Le pas de quantification est alors constant et vaut :

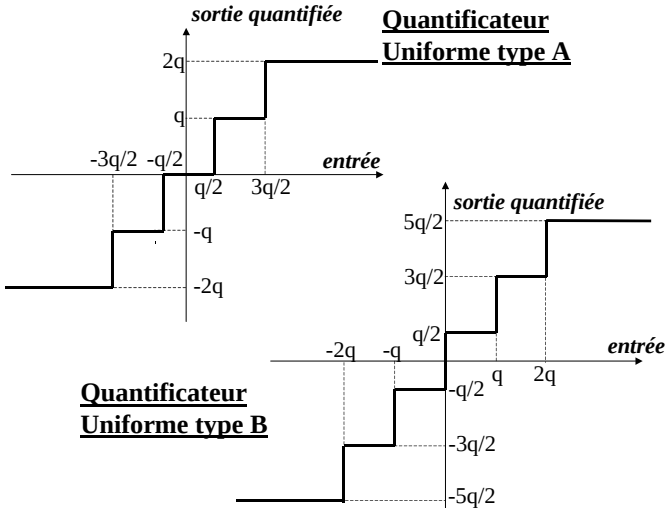
$$q = \frac{\text{pleine échelle du quantificateur}}{N}$$

On peut distinguer deux types de quantificateurs uniformes : type A et type B.

$$\textbf{type A} \quad : \quad \text{si } \left(n - \frac{1}{2}\right) q \leq s\left(t\right) \leq \left(n + \frac{1}{2}\right) q \text{ alors } s_Q\left(t\right) = nq$$

$$\textbf{type B} \quad : \quad nq \leq s\left(t\right) \leq \left(n+1\right) q \text{ alors } s_Q\left(t\right) = \left(n + \frac{1}{2}\right) q$$

Leurs caractéristiques sont données sur la figure suivante. Dans les deux cas, le domaine de quantification est symétrique par rapport à zéro.



Quantificateurs uniformes de type A et B

L'amplitude du signal d'erreur est comprise entre $-q/2$ et $q/2$. Quand les variations du signal sont grandes par rapport au pas de quantification, c'est-à-dire que la quantification est faite avec suffisamment de finesse, et lorsque le signal n'est pas écrêté, le signal d'erreur peut être supposé à distribution uniforme, de densité de probabilité :

$$\begin{aligned} p_e\left(x\right) &\approx \frac{1}{q} \qquad \text{pour } x \in [-\frac{q}{2}, \frac{q}{2}] \\ p_e\left(x\right) &\approx 0 \qquad \text{ailleurs} \end{aligned}$$

Dans ce cas, la puissance du bruit de quantification est donnée par :

$$\sigma_e^2 = \frac{q^2}{12}$$

Rapport signal à bruit de quantification

Considérons la quantification uniforme de type B. La gamme des amplitudes qu'il est possible de coder est soumise à une double limitation : vers les faibles valeurs, elle est limitée par le pas de quantification q et vers les fortes valeurs par $\frac{Nq}{2}$. Toute amplitude qui dépasse cette valeur ne peut être représentée et il y a **écrêtage du signal**. Il s'en suit une dégradation par distorsion harmonique, par exemple si le signal est sinusoïdal.

Pour définir et calculer le rapport signal à bruit de quantification, considérons un signal sinusoïdal non écrêté et occupant la pleine échelle du quantificateur.

On appelle **puissance crête** P_c d'un codeur la puissance du signal sinusoïdal ayant l'amplitude maximale A_m admissible sans écrêtage :

$$A_m = \frac{Nq}{2} \qquad P_c = \frac{1}{2}\left(\frac{Nq}{2}\right)^2$$

Le rapport signal à bruit de quantification est alors le rapport de la puissance crête et de la puissance du bruit de quantification.

$$\frac{P_c}{\sigma_e^2} = \frac{3}{2}N^2$$

soit en décibels et pour $N = 2^b$:

$$10\log_{10}\left(\frac{P_c}{\sigma_e^2}\right) = 20\log_{10} N + 1.76dB \simeq 6b + 1.76dB$$

Le rapport signal à bruit de quantification est proportionnel à N^2. Doubler N (ajouter un bit) améliore le rapport signal à bruit de $6dB$.

Quantification non uniforme : codage non linéaire suivant une loi segmentée

Le rapport signal à bruit de quantification varie fortement avec le niveau du signal dans le cas de la quantification uniforme. La quantification non uniforme vise à maintenir l'**erreur relative** de quantification constante, quelle que soit l'amplitude du signal alors que la quantification uniforme introduit une **erreur absolue** maximum de $\pm\frac{1}{2}q$.

La solution théorique est de faire varier le pas de quantification q proportionnellement à l'amplitude de l'entrée. Or ceci est incompatible avec la nécessité d'avoir un nombre fini de niveaux de quantifications. En pratique, la quantification non-uniforme utilise une quantification uniforme avec compression-extension : on réalise tout d'abord une compression de la dynamique du signal dans laquelle le signal x est transformé en un signal y, puis une quantification uniforme du signal y et finalement une extension (opération inverse de la compression) du signal quantifié.

La compression du signal x est destinée à amplifier les faibles amplitudes et à minimiser l'effet des fortes amplitudes. En pratique, deux caractérisitiques de compression (correspondant

à deux approximations de la loi logarithmique) sont normalisées au niveau international par le C.C.I.T.T. : la loi A et la loi μ.

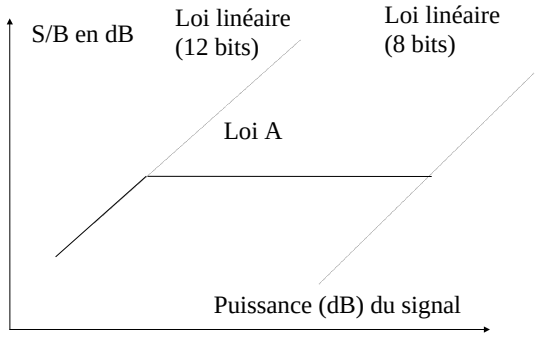
- la loi A :

$$y = sign(x)\frac{1+Ln(A|x|)}{1+Ln(A)} \quad \text{pour } \frac{1}{A} < |x| \leq 1$$
$$y = sign(x)\frac{A|x|}{1+Ln(A)} \quad \text{pour } 0 \leq |x| \leq \frac{1}{A}$$

- la loi μ :

$$y = sign(x)\frac{Ln(1+\mu|x|)}{Ln(1+\mu)} \quad \text{pour } -1 \leq x \leq 1$$

Les paramètres de A et μ déterminent l'augmentation de la dynamique du codeur. La valeur retenue pour A est 87.6, celle retenue pour μ est 255. Ces lois sont ensuite approchées par des segments de droite, de façon à coder sur un nombre fini de bits le signal ainsi distordu. La loi A est normalisée par le C.C.I.T.T. comme une loi à 13 segments alors que la loi μ est normalisée à 15 segments. La caractéristique de compression de la loi A à 13 segments est donnée sur la figure suivante. La caractéristique fait apparaître 7 segments dans le quadrant positif, on en imagine autant dans le cadrant négatif et, les deux segments entourant l'origine étant colinéaires, on obtient bien 13 segments. Les lois A et μ sont utilisées en téléphonie. Elles constituent deux approximations de la loi logarithmique proposées respectivement en Europe et aux Etats-Unis. Ces lois permettent d'obtenir un rapport signal à bruit de quantification constant à partir d'un seuil ($1/A$ ou $1/\mu$).



Rapport signal à bruit de quantification avec quantification non uniforme

L'intérêt de telles lois réside dans la diminution du nombre d'éléments binaires pour coder les échantillons, tout en conservant la dynamique du signal d'entrée (au détriment du RSB aux forts niveaux).

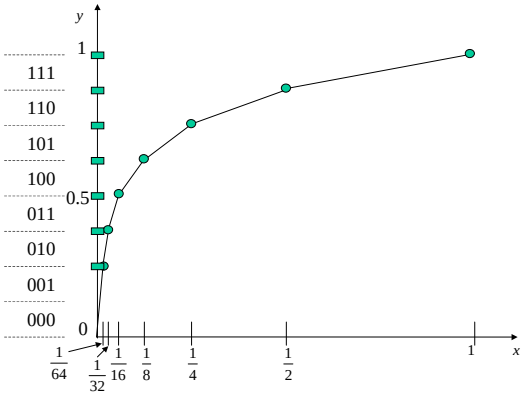
Si on prend l'exemple de la loi A, pour coder l'amplitude du signal distordu par cette loi, il faut :

- 1 bit de signe
- 3 bits indiquant le numéro de segments
- n autres bits pour coder le niveau où on se trouve dans le segment.

La dynamique du signal téléphonique est d'environ $65dB$, nécessitant 12 éléments binaires dans le cas d'un codage linéaire. Avec la compression de la loi A, le nombre de bits descend à 8, ce qui signifie que $n = 4$ bits suffisent pour un codage correct du niveau des segments.

Approximation de la loi A à l'aide de 13 segments :

$$\begin{array}{lcll}
\text{si } 0 & \leq |x| \leq \frac{1}{64} & \text{alors} & y = 16x \\
\frac{1}{64} & \leq |x| \leq \frac{1}{32} & \text{alors} & y = 8x + \frac{1}{8} \\
\frac{1}{32} & \leq |x| \leq \frac{1}{16} & \text{alors} & y = 4x + \frac{1}{4} \\
\frac{1}{16} & \leq |x| \leq \frac{1}{8} & \text{alors} & y = 2x + \frac{3}{8} \\
\frac{1}{8} & \leq |x| \leq \frac{1}{4} & \text{alors} & y = x + \frac{1}{2} \\
\frac{1}{4} & \leq |x| \leq \frac{1}{2} & \text{alors} & y = \frac{1}{2}x + \frac{5}{8} \\
\frac{1}{8} & \leq |x| \leq \frac{1}{4} & \text{alors} & y = x + \frac{1}{2} \\
\frac{1}{2} & \leq |x| \leq 1 & \text{alors} & y = \frac{1}{4}x + \frac{3}{4}
\end{array}$$



2.1.4 Annexe : Test de Kolmogorov

Le test de Kolmogorov permet de décider (avec une marge d'erreur donnée) si une variable aléatoire X suit une loi de fonction de répartition $F(x)$. Dans ce TP, il est utilisé pour l'étude de la distribution de l'erreur de quantification.

Le déroulement du test est le suivant :

1) Estimation de la fonction de répartition $\widehat{F}(x)$ à partir de K observations x_k $k = 1, ..., K$ de la variable aléatoire X. **Remarque importante** : pour se rapprocher au maximum des

conditions de validité du test de Kolmogorov, le nombre de classes pour l'estimation de la fonction de répartition doit être pris égal au nombre de points de signal.

2) Recherche de l'écart maximum $\Delta_{\max}$ entre $\widehat{F}(x)$ et $F(x)$.

3) Pour un risque de première espèce α donné (1% ou 5% généralement) vérifier que la valeur donnée dans la table de Kolmogorov est supérieure à $\Delta_{\max}$. Si c'est le cas, on décide que X suit la loi de fonction de répartition $F(x)$ avec un risque d'erreur α%. Rappelons la notion de risque lorsqu'on fait un test d'hypothèse H_0 contre H_1 : 2 risques sont définis :
$\alpha = P$[rejeter H_0 sachant H_0 vraie] risque de non détection
$\beta = P$[accepter H_0 sachant H_0 fausse] risque de fausse alarme
Pour expliquer qualitativement le rôle joué par les risques α et β, imaginons la situation suivante où on effectue la surveillance militaire d'un terrain. Prenons l'hypothèse H_0 : "*il y a des ennemis*" et l'hypothèse H_1 : "*il n'y a pas d'ennemis*". Le risque α est le risque de croire qu'il n'y pas d'ennemis et donc, de ne pas tirer alors qu'il y a effectivement des ennemis sur le terrain... Dangereux!... Le risque β est le risque de tirer en croyant qu'il y a des ennemis alors qu'il n'y a personne. Le second risque β est moins important que le premier qui peut avoir des conséquences désastreuses... Théoriquement, le résultat d'un test n'est valable que si l'on précise les valeurs des 2 risques. Le test de Kolmogorov, très simple à mettre en oeuvre, ne prend malheureusement en compte que le risque de première espèce α.

La table suivante donne l'écart maximal théorique, Δ_{kolmo}, entre fonctions de répartition empirique et théorique pour accepter l'hypothèse H_0 avec un risque α de 5 ou 1%, en fonction de Nc (nombre de points de calcul de la fonction de répartition, c'est-à-dire nombre de classes de l'histogramme). Si l'écart maximal mesuré $\Delta_{\max}$ est inférieur à Δ_{kolmo}, on accepte l'hypothèse H_0 avec un risque α. Cette table n'est valable que pour N_c inférieur à 100. Pour N_c plus grand, on sait calculer $P(\max|\widehat{F} - F| > \Delta_{kolmo})$ sous H_0 (Vladimir N.Vapnik, "The Nature of Statistical Learning Theory", Springer Ed. p87). Or :

$$\alpha = P\,[\text{rejeter } H_0 \mid H_0 \text{ vraie}] = P(\max|\widehat{F}-F| > \Delta_{kolmo}) \simeq \sum_{k=1}^{k=+\infty} 2(-1)^{k-1}\exp(-2k^2\,N_c\,\Delta_{komo}^2)$$

On mesure tout d'abord $\Delta_{\max} = \max|\widehat{F} - F|$, écart maximal entre la fonction de répartition théorique et la fonction de répartition estimée. On calcule ensuite :

$$P_{\Delta_{\max}} = P(\max|\widehat{F} - F| > \Delta_{\max}) \simeq \sum_{k=1}^{k=+\infty} 2(-1)^{k-1}\exp(-2k^2\,N_c\,\Delta_{\max}^2)$$

Or cette probabilité est une fonction décroissante de $\Delta_{\max}$. Donc si $P_{\Delta_{\max}} > \alpha$, on peut en déduire que $\Delta_{\max} < \Delta_{kolmo}$. Il n'est donc pas nécessaire de caluler Δ_{kolmo} pour prendre la décision. En effet, on accepte l'hypothèse H_0 si :

$$\sum_{k=1}^{k=+\infty} 2(-1)^{k-1}\exp(-2k^2\,N_c\,\Delta_{\max}^2) > \alpha$$

Le calcul de cette probabilité sera effectué sous Matlab, la somme infinie pouvant être approchée par une somme de 1 à K "très grand" (par exemple $K = 1000$).

N_c	$\alpha = 5\%$	$\alpha = 1\%$
5	0.5633	0.6685
10	0.4087	0.4864
15	0.3375	0.4042
20	0.2939	0.3524
25	0.2639	0.3165
30	0.2417	0.2898
40	0.2101	0.2521
50	0.1884	0.2260
60	0.1723	0.2067
70	0.1597	0.1917
80	0.1496	0.1795
90	0.1412	
100	0.1340	

2.2 Travail à réaliser

Lancer `ech_stat` sous Matlab.

2.2.1 Echantillonnage

Echantillonnage et périodisation du contenu spectral.

Générer N=100 points d'un signal sinusoïdal de fréquence 5000 Hz échantillonné à 100 000 Hz.

Dans une période du sinus, combien y a-t-il de points ?

Effectuer le "spectre" du signal (cette fonction sera étudiée en détail dans le TP Corrélations et Spectres).

Expliquer ce que l'on obtient, en particulier la présence des deux "pics" et leurs fréquences respectives (utiliser le zoom...).

Changer le nombre de points N=1000. Que se passe-t-il ? (utiliser le zoom)

Passer en fréquences normalisées (bouton "changement échelle fréquentielle"). Retrouver les valeurs des fréquences intéressantes.

Effet de la fréquence d'échantillonnage sur les représentations temporelle et spectrale d'un signal : le repliement

Reprendre le signal généré au départ : N=100 points, sinus de fréquence 5000 Hz, échantillonné à 100 000 Hz.

Pour différentes fréquences d'écantillonnage Fe, observez le signal en temporel et en spectral :

Fe= 50 000 Hz

Fe= 20 000 Hz : combien de points par période du sinus ?

Fe= Fréquence de Shannon : combien de points par période du sinus ? Que se passe-t-il du point de vue spectral ?

Fe= 7 500 Hz : sur le spectre, d'où proviennent les pics observés ? (Faire un dessin). La fréquence du sinus reconstitué est de quelle valeur ? Vérifier sur la représentation temporelle que le nombre de points par période du sinus correspond bien à cette fréquence.

Fe= 4 000 Hz : mêmes questions.

Pour un signal carré

Générer N=1000 points d'un signal carré de fréquence fondamentale de 800 Hz, échantillonné à 100 000 Hz.

Visualiser son spectre. Qu'observez-vous ?

Changer la fréquence d'échantillonnage : Fe=10 000 Hz. Observez le spectre. Que se passe-t-il ? Un tel signal est-il échantillonnable en théorie ?

Pour un fichier signal

Charger le fichier "mf_c1_v6" (fixer la fréquence d'échantillonnage à 40 kHz pour la visualisation spectrale). Ce signal correspond à une passe d'un navire (Moon Fish) au dessus d'un hydrophone de la base du Brusc (près de Toulon). Visualiser le spectre. A votre avis, le signal a-t-il été correctement échantillonné ? Pour répondre correctement à la question, il est nécessaire de visualiser ce spectre en échelle semilogarithmique.

2.2.2 Quantification

Quantification uniforme grossière

Générer N=1000 points d'un signal sinusoïdal de fréquence 20 Hz échantillonné à 2000 Hz.

Quantifier ce signal à l'aide d'un quantificateur uniforme à 4 niveaux. Pour voir l'erreur de quantification, utiliser "traitement". Que se passe-t-il en temporel ?... Bien caler la dynamique !!!

Visualiser le spectre du signal quantifié. Quel est l'effet d'une quantification grossière sur un signal sinusoïdal ?

Considérer une dynamique de 40 pour une quantification uniforme à 4 niveaux. Effectuer la quantification uniforme de type A et de type B du signal sinusoïdal précédent. Conclusions?

Quantification uniforme haute résolution

Vérifier la formule liant SNR et nombre de niveaux de quantification dans le cas haute résolution (nombre de niveaux de quantification 2^b avec $b = 16, 15, 14...$) :

$$\frac{P_c}{\sigma_e^2} = \frac{3}{2}N^2$$

Montrer en particulier que doubler N améliore le rapport signal à bruit de $6dB$.

Une des hypothèses couramment utilisée en quantification est de dire que *l'erreur de quantification est uniformément répartie sur* $[-q/2, +q/2]$.

Générer N=1000 points d'un signal sinusoïdal de fréquence 2100 Hz échantillonné à 100 000 Hz. Effectuer la quantification uniforme de ce sinus sur 14 bits (attention à la dynamique !). Observer l'erreur de quantification puis sa densité de probabilité. En appuyant sur OK, différentes réalisations du même processus aléatoires sont générées. Que penser de cette hypothèse ?

Pour s'assurer de la validité de cette hypothèse, on effectue un test de Kolmogorov. Jusqu'à quel nombre de bits de quantification peut-on considérer que cette hypothèse est valable ?

2.2.3 Restitution

Générer N=100 points d'un signal sinusoïdal de fréquence 2100 Hz, échantillonné à 100 000 Hz. Comparer les méthodes de restitution par bloqueur, interpolateur linéaire, extrapolateur et filtrage tant sur le plan temporel que sur le plan spectral.

Mêmes questions lorsque le signal sinusoïdal est échantillonné à 10 000 Hz.

Mêmes questions pour un signal carré de fréquence fondamentale 2100 Hz, échantillonné à 100 000 Hz.

2.2.4 Signal de parole

Charger le fichier signal “allo”, fichier de parole. Observer le signal en temporel, remarquer le caractère non stationnaire du signal.

Effectuer la densité de probabilité du signal et retrouver la validité apparente de l'hypothèse de loi double exponentielle souvent énoncée pour des signaux de parole.

Comparer la quantification uniforme et non uniforme (logarithmique) de ce signal à nombre de niveaux fixé à 8. En particulier, comparer la puissance de l'erreur de quantification.

2.2.5 Démonstrations

Si le TP se déroule en salle TSI (B301), des démonstrations peuvent être faite par les enseignants en traitement du signal et en traitement d'image.

Quantification des images

On utilisera le logiciel de traitement d'images et, en chargeant une image noir et blanc, on mettra en évidence l'effet de la quantification sur des images.

Quantification de la parole

Sur le poste équipé de haut-parleurs, on observera les effets de la quantification et du sous-échantillonnage sur les signaux de parole.

2.3 Elements de programmation Matlab

2.3.1 Quelques fonctions Matlab utiles dans le TP

Pour connaître le mode d'appel de ces fonctions, penser à l'aide en ligne de Matlab : `help ''nom de la fonction''` ou `lookfor ''mot''`.
`fft :` transformée de Fourier discrète
`ifft :` transformée de Fourier inverse
`interp1q:` interpolation linéaire
`quantiz:` quantification uniforme du signal.
`lin2mu:` conversion à la loi μ.
`mu2lin:` conversion inverse (de la loi μ à la loi linéaire).

2.3.2 Echantillonnage

Génération d'une sinusoïde : Pour générer le signal obtenu après l'échantillonnage d'une sinusoide de fréquence $20\,KHz$ échantillonnée à un rythme de 1 échantillon toute les 10 microsecondes, on peut procéder soit en utilisant les fréquences physiques :

```
N = 1000;
Te = 10*10^(-6);
fsin = 20*10^3;
temps = (0:N-1)*Te;
signal = sin(2*pi*fsin*temps);
```

soit en raisonnant directement en fréquences normalisées :

```
N = 1000;
Fe = 1/(10*10^(-6));
fsin_norm = 20*10^3/Fe;
indice = 0:N-1;
signal = sin(2*pi*fsin_norm*indice);
```

Visualisation de la densité spectrale de puissance : Le calcul et le tracé de la densité spectrale de puissance seront détaillés au cours du TP Corrélations et Spectres.
L'instruction de base pour réaliser la transformée de Fourier discrète d'un signal `x` est `fft(x)`.
Les commandes suivantes permettent de tracer la densité spectrale de x soit en fréquence normalisée :
`nfft = 2^nextpow2(N);` %calcul de la puissance de 2 immédiatement supérieure à N.
`plot(linspace(0,1,nfft),abs(fft(x,nfft)).^2./nfft)`.
soit en fréquence physique :
`plot(linspace(0,Fe,nfft),abs(fft(x,nfft)).^2./nfft)`.
Restitution : la commande `y = kron(x, ones(1,Nrestit));` permet de restituer par bloqueur d'ordre 0 le signal `x` échantillonné à `Te`. Le signal `y` résultant est une version suréchantillonnée de `x` de période d'échantillonnage `Te/Nrestit.`
La commande `interp1q` permet de réaliser l'interpolation linéaire (taper `help interp1q`);
La commande `interp` permet de générer un filtre passe-bas d'interpolation.

2.3.3 Quantification

Quantification uniforme : La fonction `quantiz` permet de réaliser directement la quantification sous Matlab.
Etude statitique de l'erreur de quantification : La fonction `var` donne une estimation de la variance. Pour l'estimation de la densité de probabilité et le test de Kolmogorov se reporter au TP 1.
Quantification non uniforme selon la loi μ : voir les fonctions `lin2mu, mu2lin et auwrite`.

Chapitre 3

TP Analyse Spectrale - Corrélations et Spectres

Le but de ce TP est d'analyser des estimateurs de la fonction de corrélation et de la densité spectrale de puissance (DSP).

La fonction de corrélation constitue une mesure de ressemblance entre deux signaux (intercorrélation) ou, si elle est appliquée à un seul signal, cette fonction, appelée alors autocorrélation, mesure le lien qui existe entre les différentes valeurs du signal à des instants différents. Cette quantité dépend d'un paramètre de décalage temporel (déphasage entre les deux signaux dans le cas de l'intercorrélation) et possède selon la nature des signaux, plusieurs définitions.

La densité spectrale de puissance (transformée de Fourier de la fonction d'autocorrélation) reflète la contribution qu'apporte chaque fréquence à la puissance moyenne du signal.

3.1 Rappels

3.1.1 Propriétés des fonctions de corrélation

La fonction de corrélation se définit de différentes façons suivant la classe de signaux à laquelle on s'adresse. Nous donnons ci-après les différentes définitions de la fonction d'intercorrélation. Pour la fonction d'autocorrélation, il suffit de faire $y = x$.

Signaux déterministes

- Energie finie

$$C_{xy}(\tau) = \int_{-\infty}^{+\infty} x(t)y^*(t-\tau)dt$$

- Puissance finie

$$C_{xy}(\tau) = \lim_{T \to +\infty} \frac{1}{T} \int_0^T x(t) y^*(t-\tau) dt$$

Cas particulier des signaux périodiques :

$$C_{xy}(\tau) = \frac{1}{T} \int_0^T x(t) y^*(t-\tau) dt$$

Signaux aléatoires

$$C_{xy}(\tau) = E[x(t) y^*(t-\tau)]$$

Lorsque $y = x$ on parle de fonction d'**autocorrélation** ($C_{xx}(\tau) = C_x(\tau)$).

Propriétés de la fonction d'autocorrélation

Parité : $C_x(-\tau) = C_x(\tau)$
Maximum en zéro : $|C_x(\tau)| \leq C_x(0)$
Puissance moyenne du signal $= C_x(0)$

Propriétés de la fonction d'intercorrélation

Symétrie hermitienne : $C^*_{xy}(-\tau) = C_{xy}(\tau)$
Majoration : $|C_{xy}(\tau)| \leq \frac{1}{2}(C_x(0) + C_y(0))$

3.1.2 Algorithmes de calcul des fonctions de corrélation

Pour estimer la fonction d'autocorrélation ou d'intercorrélation, on utilise deux types d'estimateurs.

Dans une première approche, la fonction d'autocorrélation est estimée comme la valeur moyenne de $x(n)x(n+k)$. Si on ne dispose que de N échantillons du signal $x(n)$ (réel), $C_{xx}(k)$ ne peut être estimée qu'à partir de $N-k$ valeurs :

$$\widehat{C}_b(k) = \frac{1}{N} \sum_{n=0}^{N-k-1} x(n)x(n+k) \quad 0 \leq k \leq N-1 \tag{3.1}$$

Lorsque k tend vers $N-1$ peu de termes interviennent dans le calcul de la moyenne alors que le terme de normalisation reste égal à $\frac{1}{N}$. Cela a pour conséquences d'introduire un **biais** dans l'estimation : la corrélation est pondérée par une fenêtre **triangulaire**. Ce premier estimateur $\widehat{C}_b(k)$ est donc biaisé.

Pour éliminer ce biais, un second estimateur peut être défini de la façon suivante :

$$\widehat{C}_{nb}(k) = \frac{1}{N-k} \sum_{n=0}^{N-k-1} x(n)x(n+k) \quad 0 \leq k \leq N-1 \tag{3.2}$$

Le choix entre ces deux estimateurs se porte normalement sur l'estimateur non biaisé mais, pour les valeurs k proches de N, la variance de cet estimateur augmente considérablement ce qui n'était pas le cas du précédent estimateur. D'autre part, il peut être intéressant de choisir l'estimateur biaisé ce qui est équivalent à prendre la fonction de corrélation multipliée par une fenêtre temporelle triangulaire.

Remarque : Le calcul de ces deux estimateursl requiert de l'ordre de $\frac{N^2}{2}$ multiplications et additions. Pour réduire le coût calculatoire, on peut utiliser un algorithme à base de transformée de Fourier rapide (FFT). En effet, les deux estimateurs ci-dessus représentent à un facteur multiplicatif près la même opération de convolution discrète $x(k) * x(-k)$. Pour calculer cette convolution, on peut utiliser le fait que

$$TF(x(k) * x(-k)) = X(f)X^*(f) = |X(f)|^2$$

avec $X(f)$ la transformée de Fourier de $x(n)$. Ainsi, les sommes temporelles coûteuses en coût calculatoire contenues dans (3.1) et (3.2) peuvent être remplacées par $FFT^{-1}\left[|FFT(x(n))|^2\right]$, ce qui rend le calcul plus rapide.

3.2 Exemples d'utilisation des fonctions de corrélation

Nous donnons ci-après deux exemples d'utilisation des fonctions de corrélation.

3.2.1 Détection d'un signal périodique noyé dans un bruit

Par détection, on entend détection de présence. Il ne s'agit pas de retrouver la forme du signal périodique mais de détecter sa présence, de savoir si ce signal existe ou non.

Soit $x(n)$ le signal périodique de période inconnue noyé dans un bruit centré $b(n)$:

$$y(n) = x(n) + b(n)$$

La fonction d'autocorrélation du signal $y(n)$ s'écrit :

$$C_{yy}(k) = C_{xx}(k) + C_{xb}(k) + C_{bx}(k) + C_{bb}(k)$$

Si le bruit $b(n)$ est indépendant du signal $x(n)$, les intercorrélations $C_{xb}(k)$ et $C_{bx}(k)$ sont nulles pour tout k. Si de plus la densité spectrale du bruit est absolument continue, on a :

$$\lim_{k \to +\infty} C_{bb}(k) = 0$$

D'où :

$$C_{yy}(k) \to C_{xx}(k) \ \ lorsque \ \ k \to +\infty$$

En calculant la fonction d'autocorrélation de $y(n)$, on va voir apparaître celle du signal périodique $x(n)$ aux grandes valeurs de k.

3.2.2 Identification d'un filtre

Considérons un filtre ou d'une façon plus générale un système linéaire invariant dans le temps, de réponse temporelle $h(t)$ et de réponse fréquentielle $H(f)$ (fonction de transfert). La relation temporelle qui relie la sortie $y(t)$ à l'entrée $x(t)$ est la suivante :

$$y(t) = x(t) * h(t) = \int_{-\infty}^{+\infty} x(u)h(t-u)du$$

On obtient la même relation sur les fonctions de corrélation :

$$C_{yx}(\tau) = C_{xx}(\tau) * h(\tau)$$

où $C_{yx}(\tau)$ est l'intercorrélation entre la sortie et l'entrée du filtre.

Si le signal $x(t)$ est un bruit blanc, c'est-à-dire $C_{xx}(\tau) = \delta(\tau)$, on a :

$$C_{yx}(\tau) = \delta(\tau) * h(\tau) = h(\tau)$$

L'intercorrélation entre la sortie et l'entrée du filtre correspond à la réponse temporelle du filtre. La transformée de Fourier de l'intercorrélation permet d'obtenir la réponse fréquentielle. Une autre possibilité consiste à estimer le spectre de la sortie du filtre :

$$Y(f) = X(f)H(f)$$

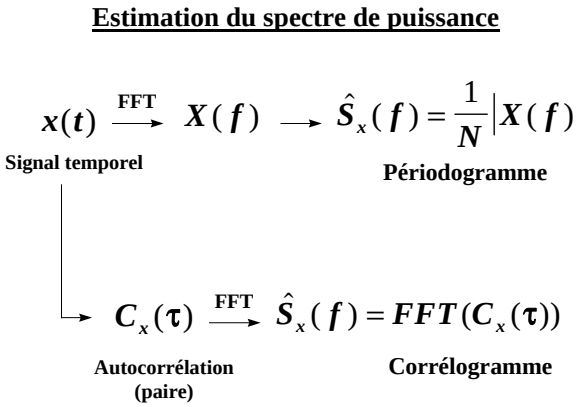
La fonction de transfert $H(f)$ permet une interprétation plus visuelle de la nature du filtre (passe-haut, passe-bas, passe-bande).

3.3 Analyse spectrale

3.3.1 Différents estimateurs

Il existe 2 grandes familles d'estimateurs de la densité spectrale de puissance (DSP) :

- les méthodes du PERIODOGRAMME
- les méthodes du CORRELOGRAMME



Ces deux catégories de méthodes mettent en oeuvre la transformée de Fourier discrète définie par :

$$X(f) = \sum_{n=0}^{N-1} x(n) e^{-i2\pi fn}$$

L'algorithme de transformée de Fourier discrète rapide, FFT, requiert un nombre de points fréquentiels N_f égal à une puissance de 2 afin d'obtenir la meilleure efficacité en termes de temps de calcul. Le spectre $S_x(f)$ est calculé aux fréquences normalisées :

$$f = \frac{k}{N_f} \quad k = 0, \cdots, N_f - 1$$

3.3.2 Intérêt du zero-padding

A titre d'exemple, la transformée de Fourier discrète (TFD) d'un sinus

$$x(n) = Acos(2\pi f_0 n + \phi) \quad n = 0, \cdots, N-1$$

donne :

$$\begin{aligned} X(f) &= \frac{A}{2}\left(e^{-i(\pi(f-f_0)(N-1)-\phi)} \frac{sin(\pi(f-f_0)N)}{sin(\pi(f-f_0))} \right. \\ &\quad \left. + e^{-i(\pi(f+f_0)(N-1)+\phi)} \frac{sin(\pi(f+f_0)N)}{sin(\pi(f+f_0))} \right) \end{aligned}$$

Ceci correspond à 2 noyaux de Dirichlet (ressemblant à 2 sinus cardinaux), l'un centré en f_0, l'autre en $-f_0$. Chaque motif correspondant à un noyau de Dirichlet se présente sous forme d'un lobe principal et de lobes secondaires, avec une pseudo période de $\frac{1}{N}$, et le résultat du calcul est représenté sur seulement N points. Ceci donne 1 point par pseudo-période et c'est trop peu pour que, visuellement, on voit clairement apparaître le noyau de Dirichlet centré sur la fréquence d'intérêt. Afin d'améliorer la visualisation du résultat de la TFD, il faut prendre

un nombre de points en fréquence N_f grand devant le nombre d'échantillons du signal et faire ainsi du "zero-padding".

3.3.3 Corrélations théoriques de signaux particuliers

- Sinusoïde à phase aléatoire

$$x(n) = Acos(2\pi fn + \phi) \rightarrow C_x(k) = \frac{A^2}{2}cos(2\pi fk) \tag{3.3}$$

- Bruit blanc

$$C_x(k) = \sigma^2\delta(k) \tag{3.4}$$

3.3.4 Périodogramme d'une sinusoïde bruitée et estimation du SNR

On considère N échantillons d'une sinusoïde de fréquence f_0 et d'amplitude A perturbée par un bruit blanc de puissance σ^2. Le périodogramme de ce signal est :

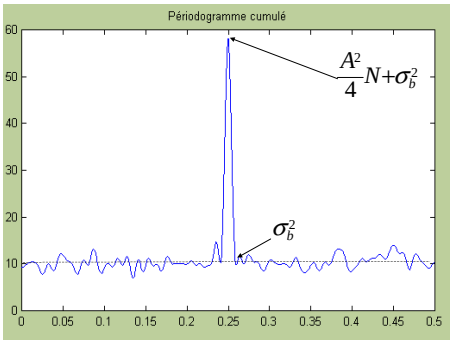
$$S_x(f) = \frac{1}{N}|X(f)|^2 \approx \sigma^2 + \frac{A^2}{4}N(sinc^2(\pi N(f - f_0)) + sinc^2(\pi N(f + f_0))) \tag{3.5}$$

avec $sinc(x) = \frac{sin(x)}{x}$.

L'amplitude du pic à la fréquence $f = f_0$ correspond donc à $\frac{A^2}{4}N + \sigma^2$ ce qui permet d'estimer le rapport signal-à-bruit (SNR) :

$$SNR = 10log\left(\frac{Puissance\ de\ la\ sinusoide = \frac{A^2}{2}}{Puissance\ du\ bruit\ = \sigma^2}\right) \tag{3.6}$$

En effet, l'observation de la DSP obtenue permet de mesurer le niveau de bruit moyen σ^2 ainsi que l'amplitude du sinus A, comme l'illustre la figure suivante.



Exemple de DSP estimée d'une sinusoïde bruitée en échelle linéaire.

3.4 Travail à effectuer

Lancer analyse_spectrale sous Matlab.

Le but de ce TP est d'effectuer de l'analyse spectrale par Transformée de Fourier (TF) et de comprendre les intérêts, inconvénients et nature des paramètres des estimateurs existants.

Charger successivement les fichiers *Eng8_ e5.txt* et *Eng8_ e11.txt* (dans le logiciel analyse_spectrale, faire "choix du signal", "fichier") : *ces signaux représentent des essais de fatigue de trains épicycloïdaux. La roue de petite vitesse est menante. Elle possède 56 dents et a une fréquence de rotation de 12,43 Hz. La roue de grande vitesse possède 15 dents et une fréquence de rotation de 46,4 Hz. On dispose de mesures issues de capteurs en position horizontale, la fréquence d'échantillonnage est de 6365 Hz. Le premier fichier correspond à des mesures effectuées le 5ème jour et le deuxième fichier à des mesures effectuées le 11ème jour. Le 5ème jour, l'état des dentures est sain tandis que le 11ème jour, il est dégradé : la dent 51 est usée à 100%, les dents 24, 26, 27, 31 et 55 sont usées à 75% et la dent 56 est usée à 15%.* La question posée est de savoir si l'analyse spectrale ou le calcul de corrélation nous permettrait de mettre en évidence l'usure des dents.

C'est pourquoi, dans ce TP, nous allons étudier différents estimateurs de la fonction d'autocorrélation et de la Densité Spectrale de Puissance (DSP). Mais avant de travailler sur des signaux réels, nous allons valider les différents estimateurs sur des signaux tests.

Remarque : *toutes les fréquences sont données en fréquences normalisées.*

Autre remarque : *il reste encore des "bugs" dans ce logiciel. En cas de comportement bizarre (refus d'afficher un signal qui devrait devoir s'afficher), utiliser le bouton "reset".*

3.4.1 Autocorrélations

Autocorrélation d'un sinus

Générer 50 échantillons d'une sinusoïde de fréquence 0.134 (phase uniformément répartie sur $[0, 2\pi]$). Observer les estimations de son autocorrélation en utilisant le bouton "ok" pour générer à chaque fois une nouvelle réalisation de signal.

Pour l'estimateur biaisé, quelle est l'allure du biais ?

Pour l'estimateur non biaisé, dans quelle partie de l'autocorrélation la variance est-elle la plus importante ? Expliquer.

Retrouver les caractéristiques du signal (puissance et fréquence).

Rappel : l'expression théorique de la corrélation est donnée par (3.3).

Autocorrélation d'un bruit blanc

Générer un bruit blanc ($N = 100$) de variance $\sigma^2 = 4$. Utiliser le bouton "ok" pour voir des réalisations différentes de ce processus aléatoire. Observer les estimations de son autocorrélation, utiliser le bouton "ok" pour visualiser différentes estimations correspondant à différentes réalisations du signal.

Des 2 estimations (biaisée ou non biaisée), laquelle paraît la plus satisfaisante ? Expliquer.

Retrouver les caractéristiques du bruit (remarquer que, pour plus de facilité, la fenêtre "Informations et mesures" fournit la valeur et l'indice du maximum de la fonction).

Rappel : l'expression théorique de la corrélation est donnée par (3.4).

Autocorrélation d'un sinus bruité

Générer 500 échantillons d'une sinusoïde bruitée (bruit blanc additif) de fréquence 0.0134 ($SNR = -7\,dB$). Pour cela, sachant que les sinusoïdes générées sont d'amplitude $A = \sqrt{2}$ (i.e. de puissance unité), régler le niveau de bruit nécessaire en utilisant la définition du SNR donnée par (3.6). Observer le signal et l'estimation biaisée de son autocorrélation (ne pas oublier d'utiliser le bouton "ok" pour générer plusieurs réalisations du processus). Montrer que l'estimation de l'autocorrélation permet de retrouver les caractéristiques du signal : présence d'un signal périodique, puissances du sinus et du bruit.

3.4.2 Estimation spectrale

Périodogramme

Générer une sinusoïde ($N = 128$, $f = 0.134$). Observer le périodogramme en utilisant une taille de FFT = 128 (en échelle linéaire et logarithmique).

Remarque : *On rappelle que pour le périodogramme, le seul paramètre à fixer est la taille de la FFT.*

Recommencer en faisant du zero-padding avec une taille de FFT de 4096. Expliquer les différences observées : l'expression théorique de la DSP est donnée par (3.5) et des rappels sur le zero-padding sont fournis au paragraphe 3.3.2). Retrouver les caractéristiques du signal : fréquence et amplitude du sinus.

Charger le fichier *quisuisje0.txt* et cliquer sur le bouton "signal" pour observer le signal en temporel. En faire l'analyse spectrale en utilisant le périodogramme : retrouver les caractéristiques du signal. Ne pas oublier d'observer l'estimateur spectral obtenu en échelle log ! Que représente la DSP à la fréquence $f = 0$?

Périodogramme modifié

Pour une sinusoïde ($N = 100$, $f = 0.25$), analyser et comparer les effets des différentes fenêtres en utilisant le périodogramme modifié. Classer-les en fonction de leur pouvoir à réduire l'amplitude des lobes secondaires (utiliser la représentation en échelle logarithmique et le bouton *hold*).

Remarque : *Pour le périodogramme modifié, les paramètres à choisir sont le type de fenêtre d'apodisation et la taille de la FFT.*

Analyse de signaux sinusoïdaux : intérêt du périodogramme modifié

Charger le fichier *quisuisje1.txt* et cliquer sur le bouton "signal" pour observer le signal en temporel. En utilisant le périodogramme modifié et les différentes fenêtres d'apodisation, donner les caractéristiques de ce signal.

Refaire la même analyse sur le fichier *quisuisje2.txt*.

Corrélogramme

Générer une sinusoïde ($N = 100$, $f = 0.134$). Observer le corrélogramme avec l'estimateur de la corrélation non biaisée en utilisant une taille de FFT = 4096. Comparer avec le corrélogramme utilisant la corrélation biaisée. Qu'en concluez-vous ? Comparer corrélogramme biaisé et périodogramme (utiliser le bouton "hold").

Remarque : *Pour le corrélogramme, les seuls paramètres à choisir sont la nature de l'estimation de la corrélation (biaisée ou non biaisée) et la taille de la FFT.*

Blackman-Tukey

Pour le même signal, utiliser le corrélogramme de Blackman-Tukey.

Remarque : *Pour la méthode de Blackman-Tukey, il est préconisé de l'utiliser sur la corrélation biaisée afin de garantir une DSP positive. Toutefois, il est intéressant de voir ce qu'on obtient en utilisant l'estimation de la corrélation non biaisée. C'est pourquoi les paramètres à choisir pour cette méthode sont la nature de l'estimation de la corrélation (biaisée ou non biaisée), le type de fenêtre d'apodisation, la taille de la fenêtre d'apodisation (**attention, appliquée sur la corrélation qui est de taille double par rapport au signal**) et la taille de la FFT.*

Faire varier tous les paramètres afin d'en observer leur effet (en particulier sur la positivité de la DSP estimée et sur les lobes secondaires des fenêtres spectrales équivalentes) : choix du type de corrélation, choix de la fenêtre, choix de la taille de la fenêtre.

Générer 1000 échantillons d'une sinusoïde bruitée (bruit blanc additif) de fréquence 0.134 ($SNR = -7\,dB$). Comparer l'analyse spectrale obtenue avec le périodogramme modifié et avec Blackman-Tukey, utilisé par exemple en prenant la corrélation biaisée et la fenêtre de Blackman (permettant d'avoir une DSP positive). Que remarquez vous sur la variance de l'estimation spectrale ?

Périodogramme de Bartlett

Générer un bruit blanc ($N = 4000$) de variance $\sigma^2 = 5$. Observer son périodogramme en utilisant une taille de FFT = 8192. Comparer le résultat obtenu à l'expression théorique de la DSP d'un bruit blanc. En remarquant que la fenêtre "informations et mesures" fournit la valeur moyenne de la DSP ainsi que sa variance, expliquer la différence entre la théorie et l'estimation obtenue de cette DSP. Ne pas hésiter à utiliser le bouton "ok" pour générer plusieurs réalisations du processus aléatoire.

Pour ce même type de signal, utiliser le périodogramme de Bartlett en choisissant des tailles de fenêtre différentes : 1000 et 100 par exemple. Qu'observe-t-on ?

Remarque : *Pour le périodogramme de Bartlett, les paramètres à choisir sont la taille de la fenêtre d'analyse et la taille de la FFT.*

L'utilisation de la méthode de Bartlett se révèle intéressante sur le bruit blanc mais elle s'accompagne d'un inconvénient. Afin de le mettre en évidence, recommencer la même expérience sur un signal constitué d'un sinus bruité ($N = 4000, f = 0.2$) en conservant la même variance de bruit. Faire varier la taille de la fenêtre d'analyse de 4000, à 1000 puis à 100. Pour ce type de signal, le logiciel n'affiche pas la valeur moyenne et la variance du spectre mais la variance peut être observée en se plaçant en échelle logarithmique. En déduire les avantages et inconvénients de la méthode de Bartlett.

Périodogramme de Welch

Pour améliorer la méthode de Bartlett, on peut utiliser le périodogramme de Welch qui permet d'utiliser une fenêtre d'analyse et un recouvrement entre les tranches.

Remarque : *Pour le périodogramme de Welch, les paramètres à choisir sont la taille et le type de fenêtre d'analyse, le recouvrement (indiqué en % ici) des tranches et la taille de la FFT.*

Générer un bruit blanc de $N = 50000$ échantillons (beaucoup d'échantillons !), de variance $\sigma^2 = 5$. Choisir une fenêtre d'analyse rectangulaire de taille 1000 et une taille de la FFT de 8192.

Pour un recouvrement de 0%, afficher plusieurs réalisations successives du périodogramme de Welch en notant la variance du spectre (ou en faire une moyenne à l'oeil !).

Refaire de même pour un recouvrement de 30%, 50% et 80%. Quel est l'intérêt du recouvrement ? Son inconvénient ?

Pour un recouvrement fixé à 30% par exemple, faire varier la nature de la fenêtre d'analyse (pas sa taille). Qu'observe-t-on ? Que gagne-t-on à utiliser des fenêtres autre que la rectangulaire ? Ne pas oublier leur inconvénient...

Résumer l'influence (*avantages* **et** *inconvénients*) des divers paramètres de la méthode de Welch (la plus utilisée) :

- taille de la FFT :
- type de fenêtre d'analyse :
- taille de la fenêtre d'analyse :
- pourcentage de recouvrement :

3.4.3 Analyse spectrale de signaux réels

Après avoir étudié les estimateurs de corrélation et de DSP, analyser les signaux d'engrenage cités au début du TP. Donner une description (sommaire) de ces signaux. A votre avis, pourrait-on distinguer ces deux signaux et donc signaler l'usure des engrenages à partir de l'analyse spectrale ?

3.5 Programmation Matlab (pour ceux qui le souhaitent)

3.5.1 Quelques fonctions Matlab utiles dans le TP

`randn :` génération d'un bruit blanc normal centré et de variance unité
`fft :` transformée de Fourier discrète
`fftshift :` recentre le spectre
`ifft :` transformée de Fourier inverse
`xcorr :` autocorrélation
`specgram :` spectrogramme (utile pour le calcul du périodogramme cumulé)
`psd :` densité spectrale de puissance Pour connaître le mode d'appel de ces fonctions, penser à l'aide en ligne de MATLAB : `help ''nom de la fonction''` ou `lookfor ''mot''`

3.5.2 Autocorrélations

Elle peut être calculée de la façon suivante :

```
for k=0:N-1,
autocorb(k+1) = fact*signal(1:N-k)*signal(k+1:N)';
end
```

avec `fact =1/N` pour l'autocorrélation biaisée ou `1/(N-k)` pour l'autocorrélation non biaisée. Il existe sous Matlab la fonction `xcorr`, avec les options `'biased'` et `'unbiased'`.
Pour la génération des différents signaux (sinus, carré, bruit blanc gaussien) voir les indications Matlab du TP1.

3.5.3 Estimations spectrales

Transformée de Fourier Discrète

L'instruction de base pour réaliser la transformée de Fourier discrète d'un signal `x` est `fft(x)`. L'algorithme de transformée de Fourier rapide est utilisé par Matlab si et seulement si la longueur du vecteur `x, Nech,` est une puissance de 2. Si ce n'est pas le cas, on peut utiliser la technique du zero-padding pour s'y ramener :

```
nfft = 2^nextpow2(Nech); %calcul de la puissance de 2 immédiatement supérieure à
Nech
fft(x,nfft);
```

Exemple : Génération de 512 points d'un sinus de fréquence normalisée $f_0 = 0.2$

```
x=sin(2*pi*f0*(0:511));
```

Calcul de sa FFT et tracé du module (fonction `abs`) de sa FFT :

$$\texttt{fft\_de\_x} = \texttt{fft}(\texttt{x}, \texttt{nfft});$$

Si l'axe des `x`, n'est pas spécifié dans la commande `plot` :

$$\texttt{plot}(\texttt{abs}(\texttt{fft\_de\_x}))$$

cet axe porte alors par défaut les indices des éléments du vecteur tracé (ici `0,1,2,... nfft-1`). Pour interpréter correctement le tracé de la Transformée de Fourier Discrète, il est nécessaire de graduer l'axe des `x`, par exemple en fréquences normalisées par rapport à la fréquence d'échantillonnage :

$\texttt{axe\_des\_x} = \texttt{linspace}(0, 1, \texttt{nfft})$

$\texttt{plot}(\texttt{axe\_des\_x}, \texttt{abs}(\texttt{fft\_de\_x}))$

L'affichage du spectre peut être également recentré autour de zéro à l'aide de la commande

`fftshift`, l'axe des `x` doit alors être modifié :

$$\texttt{plot}(\texttt{linspace}(-0.5, 0.5, \texttt{nfft}), \texttt{fftshift}(\texttt{abs}(\texttt{fft\_de\_x})))$$

Périodogramme et périodogramme de Welch

La densité spectrale de puissance (DSP) peut être estimée à l'aide du **périodogramme** :

$$\texttt{den\_puis} = \texttt{abs}(\texttt{fft}(\texttt{x}, \texttt{nfft})).\hat{}2./\texttt{Nech};$$

On peut également utiliser le **périodogramme de Welch**. Cela consiste à :
• couper le signal en tranches de `Nt` points (**Nt doit être évidemment plus faible que la longueur totale du signal**) avec recouvrement possible des tranches.
• faire une FFT de chacune des tranches (avec du zero-padding et une fenêtre d'apodisation) et prendre le module au carré de la FFT,
• faire la moyenne de toutes les FFT en module au carré
Ces opérations peuvent être réalisées avec la fonction `pwelch` (voir le help sous Matlab).

Corrélogramme

Il suffit d'utiliser successivement `xcorr` et `fft`.

Chapitre 4

Filtrage Numérique

Ce TP est consacré à l'étude des filtres à Réponse Impulsionnelle Finie (RIF) et des Filtres à Réponse Impulsionnelle Infinie (RII). Dans le cas des filtres RIF, tout échantillon du signal en sortie est la somme pondérée d'échantillons du signal en entrée. Les filtres RIF sont fréquemment désignés par le terme de filtres non-récursifs, car ils ne présentent pas de boucle de réaction de la sortie vers l'entrée. Ils peuvent être synthétisés directement par un développement en série de Fourier du gabarit idéal. Le résultat obtenu peut être ensuite optimisé grâce à la méthode des moindres carrés ou à l'algorithme de Remez. La deuxième partie du TP est consacrée à l'étude des filtres RII ou filtres récursifs. La synthèse de ces filtres s'appuie sur les fonctions modèles du filtrage analogique (Tchebychev, Butterworth,...) par l'intermédiaire de la transformée bilinéaire, transformation conforme permettant de passer du plan numérique au plan analogique.

Le but du TP est de comparer pour un gabarit donné les résultats des différentes synthèses.

4.1 Filtre à Réponse Impulsionnelle Finie (RIF) : rappels

4.1.1 Définition

Ce sont des systèmes à réponse impulsionnelle finie, de fonction de transfert $H(z)$, dont les coefficients $h(k)$ sont tels que :

$$h(k) \neq 0 \text{ pour } k \in [0, N-1]$$
$$h(k) = 0 \text{ sinon}$$

On obtient l'expression de la fonction de transfert dans le plan des z :

$$H(z) = \sum_{k=0}^{N-1} h(k) z^{-k}$$

Le caractère non récursif apparaît clairement sur l'équation de récurrence liant l'entrée et la sortie du filtre :

$$y(n) = \sum_{k=0}^{N-1} h(k)\, x(n-k)$$

La méthode de synthèse des filtres RIF permet d'assurer une phase linéaire, caractéristique recherchée dans de nombreuses applications. Ceci implique une symétrie de la réponse impulsionnelle. En effet, on veut :

$$H(f) = R(f)\, e^{j\Phi(f)}$$

avec $R(f) \in \mathbb{R}$ et $\Phi(f) = -2\pi f \tau_{pg}$. La réponse impulsionnelle d'un tel filtre s'écrit :

$$h(t) = \int_{-\infty}^{+\infty} R(f)\, e^{j\Phi(f)} e^{j2\pi ft} df = \int_{-\infty}^{+\infty} R(f)\, e^{j2\pi f(t-\tau_{pg})} df$$

On décompose $R(f)$ en la somme d'une partie paire $R_p(f)$ et d'une partie impaire $R_i(f)$. La réponse impulsionnelle $h(t)$ étant réelle (dans le cas d'un signal réel le spectre d'amplitude est pair et la phase est impaire), on a :

$$h(\tau_{pg} + t) = 2\int_{-\infty}^{+\infty} R_p(f) \cos(2\pi ft)\, df = h(\tau_{pg} - t)$$

Cette relation fait donc apparaître la symétrie de la réponse impulsionnelle par rapport au point $t = \tau_{pg}$ de l'axe des temps.

4.1.2 Synthèse par la méthode de la fenêtre

On se donne un gabarit fréquentiel $H(f)$ à respecter. On calcule la réponse impulsionnelle du filtre recherché par transformée de Fourier inverse de ce gabarit. Il s'agit alors de faire une troncature afin de garder un nombre fini N d'éléments qui seront les coefficients du filtre, puis d'effectuer un décalage afin de rendre le filtre causal c'est-à-dire physiquement réalisable.

Etapes successives de la synthèse :

1) Définir un gabarit fréquentiel en fréquences normalisées

2) En faire un développement en série de Fourier

$$h(k) = \int_{-\frac{1}{2}}^{\frac{1}{2}} H\left(e^{j2\pi\widetilde{f}}\right) e^{j2\pi\widetilde{f}k} d\widetilde{f}$$

3) Multiplication par une fenêtre temporelle $W(k)$ de longueur N (ordre du filtre) avec $W(k) = 0$ pour $|k| > N$,

4) Réaliser un décalage (translation) afin de satisfaire la condition de causalité. La valeur du temps de propagation de groupe est :

$$\tau_{pg} = -\frac{1}{2\pi}\frac{d\Phi\left(\widetilde{f}\right)}{d\widetilde{f}} = \frac{N-1}{2}$$

4.1.3 Optimisation de la synthèse obtenue

Uen fois la synthèse obtenue par la méthode classique de synthèse RIF par fenêtre, on peut optimiser le filtre obtenu.

La première méthode d'optimisation consiste à minimiser au sens des moindres carrés la distance entre le gabarit $H(f)$ désiré et le gabarit du filtre obtenu par la méthode précédente. L'objectif de la seconde méthode d'optimisation est d'obtenir la meilleure approximation du gabarit $H(f)$ présentant des ondulations d'amplitude constante. Elle utilise une technique itérative : l'algorithme de Remez.

4.2 Filtre à Réponse Impulsionnelle Infinie (RII) : rappels

4.2.1 Définition

Ces systèmes sont caractérisés par des réponses impulsionnelles de durée infinie : les coefficients $h(k)$ sont non nuls sur l'intervalle $[0, +\infty[$. Ceci est réalisé par la présence de pôles dans la fonction de transfert du filtre :

$$H(z) = \frac{\sum_{k=0}^{p-1} b_k z^{-k}}{\sum_{i=0}^{m-1} a_i z^{-i}}$$

Cela se traduit par l'équation suivante :

$$y(n) = -\sum_{i=1}^{m-1} a_i y(n-i) + \sum_{k=0}^{p-1} b_k x(n-k)$$

La condition de stabilité impose que les pôles de $H(z)$ soient à l'intérieur du cercle unité. La réponse impulsionnelle infinie permet d'obtenir un filtrage plus sélectif qu'un filtre RIF pour une quantité de calcul inférieure. La linéarité de la phase est, en théorie, impossible. Cependant on peut l'obtenir approximativement dans une bande limitée.

Synthèse

On calcule tout d'abord le filtre analogique passe-bas qui lui correspond, puis on synthétise ce filtre analogique. Enfin, on revient au filtre numérique par la transformation bilinéaire. Les méthodes classiques mettent en oeuvre des transformations à partir des équations de filtres analogiques connus (Butterworth, Tchebychev...). Le passage entre les différents types de filtre (Passe-Bas, Passe-Haut, Passe-Bande et Coupe-Bande) peut se faire dans le plan des p

(analogiques) ou dans le plan des z au cours de la transformation $p \longrightarrow z$. La méthode utilisée, la transformation bilinéaire, fait correspondre le plan des z au plan des p suivant la relation :

$$p = \frac{2}{T_e}\frac{1 - z^{-1}}{1 + z^{-1}}$$

La synthèse d'un filtre RII se fait donc selon les étapes suivantes :
1) Synthèse du filtre analogique passe-bas de pulsation de coupure ω_c :

$$\text{Butterworth d'ordre } n \quad : \qquad |H(\omega)|^2 = \frac{1}{1 + \left(\frac{\omega}{\omega_c}\right)^{2n}}$$

$$\text{Tchebychev d'ordre } n \quad : \qquad |H(\omega)|^2 = \frac{1}{1 + \varepsilon^2 T_n^2(\omega)}$$

$$\text{avec } \begin{cases} T_n(\omega) = \cos(n Ar\cos(\omega)) \text{ si } |\omega| \leq 1 \\ T_n(\omega) = ch(n Arch(\omega)) \text{ si } |\omega| > 1 \end{cases}$$

2) Transformations fréquentielles en vue d'obtenir les diverses familles de filtres : passe-haut, passe-bande, coupe bande.
3) Transformée bilinéaire du filtre obtenu en 2).
4) Calcul de la fonction de transfert à partir des coefficients.
5) Visualisation : DSP, temps de propagation de groupe, réponse impulsionnelle, réponse indicielle. Remarque : la transformation bilinéaire introduit une déformation des fréquences :

$$f_\alpha = \frac{F_e}{\pi}\tan\left(\pi\frac{f_n}{F_e}\right)$$

avec f_α fréquence analogique, f_n fréquence numérique et F_e la fréquence d'échantillonnage.

4.3 Travail à effectuer

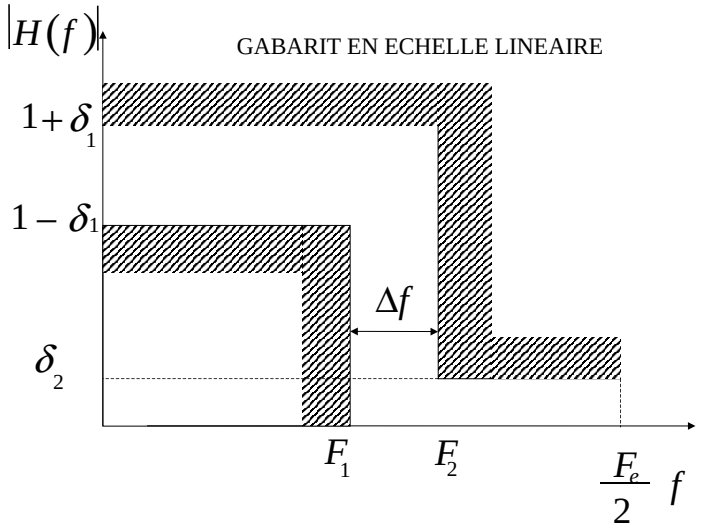
Lancer `filtnum` sous Matlab.

4.3.1 Introduction

Le TP permet de synthétiser des filtres numériques RIF et RII. Les filtres RIF peuvent être calculés par un développement en série de Fourier avec différentes fenêtres (rectangulaire, triangulaire, de Hamming et de Kaiser) et par deux méthodes d'optimisation (par les moindres carrés et par Remez). Les méthodes proposées pour les filtres RII utilisent les équations donnant des filtres analogiques connus (de Butterworth, de Chebychev I, de Chebyshev II et elliptique). L'intérêt sera porté tout d'abord sur les caractéristiques des filtres RIF, puis sur celles des filtres RII. Chacune de ces études sera suivie par des exemples de filtrage de signaux réels.

4.3.2 Gabarit

Le gabarit du filtre est défini par l'utilisateur parmi les quatre catégories suivantes : filtre passe-bas, filtre passe-haut, filtre coupe-bande et filtre passe-bande. On peut régler F_e la fréquence d'échantillonnage, Δf la largeur de la bande de transition, δ_1 l'amplitude des ondulations en bande passante et δ_2 l'amplitude des oscillations en bande atténuée. Ces amplitudes sont exprimées en échelle linéaire. Il est également possible d'utiliser comme paramètres l'ondulation en bande passante et en bande affaiblie exprimées en dB : $dp = 20\log_{10}\left(\frac{1+\delta_1}{1-\delta_1}\right)$ et $da = 20\log_{10}\frac{1}{\delta_2}$.



Choisir un gabarit en entrant ces différents paramètres. Le but du TP va être de comparer les résultats de la synthèse des différents filtres pour ce gabarit.

4.3.3 Filtres à Réponse Impulsionnelle Finie (RIF)

Remarque préalable : l'utilisateur du logiciel peut choisir l'ordre du filtre à synthétiser. Notons que pour un ordre 2, la synthèse par la fenêtre triangulaire ne donne rien (problème de définition de fenêtre) et que la synthèse par Remez "plante", l'algorithme itératif associé nécessitant un ordre minimal de 3.

Synthèse par la méthode de la fenêtre

Evaluation de l'ordre : Possédant les caractéristiques du gabarit à respecter, il est possible de calculer quel devra être l'ordre N (très approximatif et en général sous-estimé) :

$$\widehat{N} = \frac{2}{3}\log_{10}\left(\frac{1}{10\delta_1\delta_2}\right)\frac{F_e}{\Delta f}$$

où F_e est la fréquence d'échantillonnage, Δf la largeur de la bande de transition, δ_1 l'amplitude des ondulations en bande passante et δ_2 l'amplitude des oscillations en bande atténuée.

Calculer l'ordre donné par l'approximation pour le gabarit choisi.

Influence de la fenêtre : Synthétiser le filtre RIF à cet ordre $\widehat{N}$ en utilisant la méthode de synthèse RIF par fenêtre. Observer les fonctions de transfert obtenues par les quatre types de fenêtres disponibles : valeur de la pente, position et amplitude du premier lobe d'oscillation. Cette comparaison sera faite uniquement d'un point de vue qualitatif, en utilisant le bouton "hold" (et ne pas hésiter à aussi à utiliser le bouton "Ylog"). Quels sont les avantages et les inconvénients de chacune des fenêtres ? Observer le temps de propagation de groupe.

Influence de l'ordre : On appelle ordre optimal N_{opt} d'un filtre numérique, l'ordre minimal tel que le gabarit fréquentiel soit respecté. Synthétiser le filtre RIF à l'aide de la fenêtre rectangulaire en faisant varier l'ordre du filtre depuis la valeur calculée $\widehat{N}$ ($\widehat{N} < N_{opt}$!) et en augmentant progressivement jusqu'à dépasser N_{opt}. Mesurer l'influence de l'ordre sur les paramètres suivants :

• la raideur de la pente

• la position de la fréquence de coupure à $-3dB$.(on visualise $|H(f)|^2(f)$ donc la fréquence de coupure correspond à un module carré égal à 0.5)

• la pseudo-période des lobes d'oscillation

• l'allure du temps de propagation

Comparer l'amplitude des ondulations en bande passante et en bande affaiblie. Quelle remarque peut-on faire sur l'amplitude des oscillations ?

Réalisation d'un filtre particulier

Pourrait-on réaliser un filtre passe-tout? Comment faudrait-il modifier le gabarit pour synthétiser un tel filtre ? Quel serait son intérêt?

Optimisation du filtre RIF

La méthode des moindres carrés permet d'agir différemment sur les ondulations en bande passante et en bande affaiblie. Comment? Quelle propriété spécifique possède l'amplitude des oscillations lorsqu'on applique la méthode de Remez ? Observer le temps de propagation de groupe.

Réponse impulsionnelle et réponse indicielle

Observer la réponse impulsionnelle d'un filtre RIF. Comment retrouve-t-on les coefficients du filtre à partir de la réponse impulsionnelle ? Justifier son aspect symétrique. Retrouver le temps de propagation. Quelles remarques peut-on faire sur la réponse indicielle (allure générale, nombre d'oscillations en fonction de l'ordre, ...) ?

Filtrage de signaux

Vérifier la nature du filtre en lui imposant en entrée un bruit blanc (utiliser le bouton "DSP" qui calcule les DSP de l'entrée et de la sortie du filtre).

Filtrer un signal sinusoidal. Qu'observe-t-on sur la sortie ?

Filtrer une somme de 2 sinus, composée d'un sinus dans la bande passante du filtre et d'un autre sinus dans la bande atténuée. Qu'observe-t-on sur la sortie ?

4.3.4 Filtre à Réponse Impulsionnelle Infinie (RII)

A la différence des filtres RIF, on peut calculer de manière assez précise l'ordre minimal nécessaire pour un type de filtre analogique donné permettant de respecter le gabarit. Ces calculs sont faits pour chaque type de filtre dans le logiciel par le bouton "Auto".

Synthèse avec un Butterworth

Faire varier l'ordre du filtre synthétisé en démarrant en dessous de l'ordre donné par la formule (bouton "auto") et en augmentant progressivement, jusqu'à dépasser l'ordre "auto". Observer l'influence de l'ordre sur la pente et sur le temps de propagation de groupe. Comment évoluent, pour le temps de propagation, la valeur des maxima et les fréquences correspondant à ces maxima? *Pourquoi peut-il être très génant d'avoir des temps de propagation très différents dans la bande passante?*

Comparaison entre les différents filtres RII

Observer les fonctions de transfert obtenues par les quatre méthodes proposées (Butterworth, Chebyshev I et de Chebychev II et elliptique) : valeur de la pente, position du premier lobe d'oscillation, amplitude du lobe secondaire.

Quelle est la différence entre les méthodes de Chebyshev I et de Chebychev II ?

Quel est l'intérêt de la synthèse d'un filtre elliptique ?

Comparer les temps de propagation de groupe des différents filtres en ne tenant pas compte des "glitchs" obtenus par la fonction matlab (grpdelay calcule le temps de propagation de groupe à partir des coefficients des filtres mais présente quelques bugs). Quelles conséquences cela va-t-il entraîner ?

Réponse impulsionnelle

Quelle est la différence par rapport à la réponse impulsionnelle d'un RIF ?

Filtrage d'un signal

Vérifier la nature du filtre en lui imposant en entrée un bruit blanc.

4.3.5 Conclusion

Donner les avantages et les inconvénients respectifs des filtres RIF et des filtres RII.

Chapitre 5

Initiation Matlab

Matlab, abréviation de MATrix LABoratory, est un langage très performant pour le calcul numérique. Les utilisations classiques sont : le calcul mathématique, le développement d'algorithmes, la modélisation et la simulation, l'analyse de données et la visualisation, le développement d'applications telles que les interfaces utilisateur.

5.1 Commandes d'aide

Deux instructions seront très utiles par la suite :
• `lookfor ''mot_clé''` qui permet de trouver une commande à l'aide de mots clés. Exemple : `lookfor filter`. Matlab donnera en réponse la liste des fonctions intégrées dont l'aide associée comporte ce mot clé.
• `help ''commande''` donne une aide sur la commande en question. Contrairement à `lookfor`, il est nécessaire de connaître le nom exact de la commande. Exemple : `help mean.`
• `help ''toolbox''` donne toutes les commandes de la toolbox par catégories. Exemple : `help signal; help stats; help optim; help nnet;`

5.2 Matrices

5.2.1 Construction

Matlab travaille essentiellement sur des matrices, un vecteur ou un scalaire étant considéré comme une matrice avec une ligne et/ou une colonne. La saisie d'une matrice peut s'effectuer de différentes façons : soit en la rentrant comme une liste d'éléments, soit en la générant dynamiquement à l'aide de fonctions, soit encore en la chargeant depuis un fichier. **La construc-**

tion élément par élément peut se faire de la façon suivante :

$$\texttt{A} = [\texttt{1 2 3; 4 5 6; 7 8 9}]$$

Cette instruction crée une matrice 3×3 et l'affecte à la variable A. Les éléments d'une ligne sont séparés par des espaces ou des virgules et les lignes sont séparées par des point-virgules. Matlab traite aussi les matrices complexes :

$$\begin{aligned} \texttt{A} &= [\texttt{1 2; 3 4}] + \texttt{i} * [\texttt{5 6; 7 8}] \\ \texttt{A} &= [\texttt{1} + \texttt{5} * \texttt{i} \quad \texttt{2} + \texttt{6} * \texttt{i}; \ \texttt{3} + \texttt{7} * \texttt{i} \quad \texttt{4} + \texttt{8} * \texttt{i}] \end{aligned}$$

`i` ou `j` peuvent être utilisés indifféremment pour la notation imaginaire. Dans le cas où ils seraient utilisés comme variables, on peut les recréer à l'aide de :

$$\texttt{ii} = \texttt{sqrt}(-1)$$

Les éléments d'une matrice sont référencés de la façon suivante :
• `A(2,3)` est l'élément de la deuxième ligne et troisième colonne de la matrice A.
• si l'on définit un vecteur B, alors `B(4)` est le quatrième élément de ce vecteur, indifféremment qu'il soit ligne ou colonne.
• les indices des vecteurs et matrices débutent à 1 et non à 0. On peut affecter une valeur scalaire à un élément d'une matrice comme suit :

$$\texttt{A}(2,4) = 8$$

qui va affecter la valeur 8 à l'élément de la deuxième ligne et quatrième colonne de A. **Certaines fonctions permettent de créer des matrices.** Les fonctions matricielles les plus couramment utilisées sont :

`zeros`	matrice de zéros
`ones`	matrice de uns
`eye`	matrice identité
`linspace`	vecteur en progression arithmétique
`logspace`	vecteur en progression logarithmique
`diag`	création ou extraction de la diagonale d'une matrice

Par exemple, `zeros(m,n)` crée une matrice $m \times n$ remplie de zéros, et `zeros(n)` crée une matrice remplie de zéros et de taille $n \times n$. Si x est un vecteur, `diag(x)` est la matrice diagonale avec x sur la diagonale, les autres éléments étant nuls. Si A est une matrice carrée, `diag(A)` est un vecteur composé de la diagonale de A. Les matrices peuvent être construites par blocs. Soit A une matrice 3×3 :

$$\texttt{B} = [\texttt{A}, \ \texttt{zeros}(3,2); \ \texttt{zeros}(2,3), \ \texttt{eye}(2)]$$

va construire une matrice 5×5. **Les générateurs de nombres aléatoires** (taper `help stats` pour en avoir la liste) permettent de générer des matrices dont les éléments suivent une loi donnée. La commande `rand(n)`, par exemple, crée une matrice $n \times n$ dont les éléments sont uniformément distribués entre 0 et 1 ; `rand(m,n)` crée une matrice de taille $m \times n$ suivant la même distribution. **La notation ":"** permet d'exploiter la structure matricielle de Matlab et de réaliser l'économie de boucles par rapport à la plupart des langages de programmation. Par exemple :

`1:5` est le vecteur `[1 2 3 4 5]`.

Bien sûr, les incréments peuvent ne pas être entiers, la syntaxe est la suivante :

`début : pas : fin` (le pas étant par défaut égal à 1).

Exemple :

`0.2:0.2:1.2` est le vecteur `[0.2 0.4 0.6 0.8 1.0 1.2]`,

et `5:-1:1` est `[5 4 3 2 1]`.

L'exemple suivant crée un signal sinusoïdal :

```
temps = [0:0.1:2]';
y = sin(2*pi*f0*temps);
```

La notation ":" permet aussi d'accéder aux sous-matrices d'une matrice. Par exemple, `A(1:4,3)` est le vecteur colonne composé des 4 premiers éléments de la $3^{\text{ème}}$ colonne de la matrice A. `A(:,3)` est la troisième colonne de la matrice A et `A(1:4,:)` est composée des quatre premières lignes de la matrice A. Cette notation permet aussi de remplacer des morceaux de matrices : `A(:,[2 4 5]) = B(:,1:3)` remplace les colonnes 2, 4 et 5 de A par les trois premières colonnes de B. La fonction `reshape(A,m,n)` reconstruit la matrice A avec m lignes et n colonnes.

5.2.2 Opérations élémentaires

Les opérations suivantes sont disponibles dans Matlab :

$+$	addition	*conj*	conjugué uniquement
$-$	soustraction	$.'$	transposé uniquement
$*$	multiplication	$\backslash$	division à gauche
$\hat{}$	puissance	$/$	division à droite
$'$	conjugué transposé		

Ces commandes s'appliquent aussi aux scalaires. Si les tailles des matrices sont incompatibles, alors un message d'erreur apparaît, sauf dans le cas d'une opération entre un scalaire et une matrice où l'opérateur va s'appliquer au scalaire et aux éléments de la matrice. **Attention :** La division matricielle est particulière. Soit A une matrice et b un vecteur respectivement colonne ou ligne, alors :

`x=A\b` est la solution de $A * x = b$, et

`x=A/b` est la solution de $x * A = b$.
Les divisions à droite et à gauche sont liées par $b/A = (A'\backslash b')'$. Pour résoudre le système, Matlab calcule l'inverse de A quand A est inversible et sa pseudo-inverse lorsque A est singulière sans pour autant vous en avertir.

5.2.3 Opérations élément par élément

Les opérations précédentes sont les opérations matricielles classiques. Les opérations *, ^, \, / peuvent aussi s'appliquer élément par élément sur une matrice ou un vecteur en les faisant précéder d'un point. Par exemple : `[1 2 3 4].*[1 2 3 4]` ou `[1 2 3 4].^2` donnent `[1 4 9 16]`

5.2.4 Fonctions élémentaires

Les fonctions élémentaires disponibles sous Matlab, s'appliquent aussi bien à des matrices qu'à des scalaires. Les plus courantes sont :
`sin`, `cos`, `tan`, `asin`, `acos`, `atan`
`sinh`, `cosh`, `tanh`, `asinh`, `acosh`, `atanh`
`exp`, `log`, `log10`
`rem`, `abs`, `angle`, `real`, `imag`, `conj`, `sqrt`, `sign`
`round`, `floor`, `ceil`, `fix`

D'autres fonctions renvoient un scalaire lorsqu'elles s'appliquent sur des vecteurs. Dans le cas où on les appliquent à des matrices, elles agissent généralement sur chacune des colonnes indépendamment, le résultat est alors un vecteur. Ce sont :
`max`, `min`,
`sort`,
`sum`, `prod`, `median`, `mean`, `std`
`any`, `all`

Par exemple, l'élément maximal d'une matrice A est donné par `max(max(A))`, et non pas par `max(A)` qui renvoie un vecteur (consulter l'aide).

5.2.5 Fonctions matricielles

Les plus courantes sont :

`eig`	valeurs propres
`poly`	polynome caractéristique
`det`	déterminant
`inv`	inverse
`expm`	matrice exponentielle
`sqrtm`	matrice racine carrée
`size`	taille d'une matrice

5.3 Gestion de l'espace de travail

Les fonctions suivantes permettent d'avoir une vue globale de l'espace de travail :

`who`	liste les variables
`whos`	liste les variables, leur taille et leur type
`what`	liste les fichiers d'extension .m et .mat

Les variables peuvent être supprimées à l'aide de la commande `clear nom_variable`.
Attention à la commande `clear` qui utilisée seule efface toutes les variables de l'espace de travail.

Lorsque l'on quitte Matlab, toutes les variables sont perdues. On peut les sauver avec la commande `save nom_fichier nom_des_variables_a_sauver.` Le format du fichier est alors en `*.mat`. Lorsque l'on relance Matlab, on peut les recharger avec `load nom_fichier`.

Certaines commandes DOS ou Unix classiques sont valables dans Matlab :

`pwd`	indique le chemin du répertoire courant
`cd`	change de répertoire
`dir`	liste tous les éléments d'un répertoire
`what`	liste les fichiers .m, .mat et .mex d'un répertoire
`path`	obtention ou initialisation du chemin

D'autres commandes DOS ou Unix peuvent être exécutées si elles sont précédées de "!". Exemple : `!del nom_fichier.`

5.4 Boucles, tests et relations

Dans Matlab, ce type de fonctions opére quasiment de la même façon que pour les langages de type Pascal ou C. **Mais attention** : Remplacer une boucle par un traitement matriciel, lorsque cela est possible, conduit à un gain de temps d'exécution très important.

5.4.1 Boucles

L'exécution répétitive d'une suite d'instruction peut-être réalisée à l'aide de `for` ou `while`.

La syntaxe de la boucle for est la suivante :

```
for condition
        instructions
    end
```

Par exemple, soit `n` donné :

```
x = [ ];          %crée une matrice vide
for (i = 1:n)
        x = [x, i^2]
end
```

La syntaxe de la boucle while est la suivante :

```
while expression
        traitement
end
```

Le traitement va être répété tant que l'expression est vraie. Les groupes d'instructions qui vont être exécutées ou non selon la condition, peuvent être délimitées par les instruction `case` et `otherwise`.

5.4.2 Test : instruction if

Cette instruction permet d'exécuter une suite d'instructions conditionnelles. Sa syntaxe est la suivante :

```
if condition
        instructions
end
```

Le traitement va être exécuté uniquement si la condition est vraie. Des branchements multiples sont aussi possibles :

```
if condition1
        instructions1
elseif condition2
        instructions
else
        instructions
end
```

5.4.3 Relations

Les principaux opérateurs relationnels sont les suivants :

<	inférieur	>=	supérieur ou égal
>	supérieur	==	égal (“ = ” est une affectation)
<=	inférieurou égal	~=	différent

Ils peuvent être combinés avec des opérateurs logiques suivants :

& et, | ou, ~ non

Lorsque une relation est appliquée à des scalaires, le résultat est alors égal à 1 ou 0 selon qu'elle soit vraie ou fausse et lorsqu'elle est appliquée à des matrices ou des vecteurs de même taille, le résultat est une matrice dont les éléments prendront les valeurs 1 ou 0.

Une relation entre matrices est interprétée par `while` et `if` comme vraie si elle est vérifiée par tous les éléments de la matrice. Par conséquent, si l'on veut exécuter un traitement quand des matrices A et B sont égales, on peut faire :

```
if A == B
    traitement
end
```

tandis que si on veut l'exécuter lorsqu'elles ne sont pas égales, on peut faire :

```
if any(any(A~= B))
    traitement
end
```

ou tout simplement

```
if A == B else
    traitement
end
```

Il faut noter que :

```
if A~= B
    traitement
end
```

ne va pas donner le résultat escompté, car le traitement va être exécuté seulement si **tous** les éléments de A et B sont différents.

Les fonctions `any` et `all` permettent de réduire les relations entre matrices à des vecteurs ou des scalaires. Consulter l'aide à leur sujet.

Voir aussi la fonction `find` qui renvoie l'indice des éléments obéïssant à une condition. Par exemple, soit y un vecteur, alors `find(y>2)` va renvoyer le rang de tous les éléments de y qui sont strictement supérieurs à 2.

5.4.4 Pause et break

Lors de l'exécution d'un programme, on peut, à l'aide de l'instruction `pause`, suspendre son exécution et la relancer si l'utilisateur appuie sur une touche.

L'instruction `break` arrête l'exécution d'une boucle `while` ou `for`.

5.5 Fichiers .m

Matlab peut exécuter un ensemble d'instructions contenues dans un programme. Ces programmes ont une extension `.m`. Il y a deux types de programmes `.m` : les scripts et les fonctions.
`%` permet d'insérer des commentaires.
Si le dernier caractère d'un traitement est un point virgule, alors l'affichage sur la fenêtre de commande Matlab est supprimé.

5.5.1 Scripts

Un fichier script est composé d'un ensemble d'instructions Matlab. Si le fichier a le nom `prog.m`, alors la commande `prog` va l'exécuter. Les variables d'un programme script sont globales, et son exécution va changer leurs valeurs dans l'espace de travail.

5.5.2 Fonctions

Les variables d'une fonction sont locales, mais peuvent aussi être déclarées comme globales. Soit la fonction :

```
function y = moyenne(x)
             %calcul de la valeur moyenne.
         y = sum(x)/length(x);
```

Cette fonction va calculer la moyenne des éléments d'un vecteur. Dans la ligne de commande, on peut alors écrire :

```
vect = [1 2 3 4 5 6];
 moy = moyenne(vect)
```

qui va renvoyer la valeur moyenne des éléments du vecteur *vect*.

La première ligne du script d'une fonction doit obligatoirement être de la forme :

`function` $[resultat1, resultat2, ...] =$ *nom_fonction*(*argument1, argument2,...*)

5.5.3 Textes, entrées, messages d'erreurs

Les textes doivent être encadrés par des quotes : `s = 'texte'.`
Ils peuvent être affichés sur la fenêtre de commande Matlab : `disp('affichage')`.
Les messages d'erreurs peuvent être affichés à l'aide de la commande `error` qui arrête aussitôt l'exécution du programme : `error('desole!')`.
On peut aussi saisir des paramètres lors de l'exécution : `iter = input('nombre d iterations? ')`

5.5.4 Vectorisation et pré-allocation

Dans le but d'accélérer l'exécution de programmes, il est important de vectoriser les algorithmes dans les fichiers `.m`. En effet, là où d'autres langages utilisent des boucles, Matlab peut utiliser des opérations vectorielles ou matricielles. Considérons l'exemple suivant :

```
for t = 1:N
        y(t) = sin(2*pi*f0*t)
end
```

Si on vectorise ce programme, il suffit d'écrire:

```
t = 1:N;
y = sin(2*pi*f0*t);
```

Dans le cas où il n'est pas possible de vectoriser, il est possible de rendre les boucles `for` plus rapides en pré-allouant des vecteurs ou des matrices dans lesquels les résultats seront stockés. Par exemple :

```
y = zeros(1,100);
for (i = 1:100)
        y(i) = det(X^i);
end
```

En effet, si on ne pré-alloue pas un vecteur, Matlab va réactualiser la taille du vecteur `y` à chaque boucle d'itération, ce qui prend énormément de temps.

5.6 Graphiques

Matlab permet de réaliser des graphiques 2-D et 3-D. Pour avoir un aperçu des capacités de Matlab en matière de graphismes, utiliser la commande `demo`. Dans les travaux pratiques, nous utiliserons essentiellement des tracés 2-D. Soit deux vecteurs x et y de même taille. La commande

`plot(x,y)` va tracer y en fonction de x. Soit à tracer la fonction sinus de 0 à 4 :

```
temps = 0 : 0.01 : 4;
sinus = sin(2 * pi * f0 * temps); plot(temps,sinus);
```

Si l'on souhaite tracer un deuxième graphique sur une figure différente, taper `figure` ou `figure(i + 1)` si la figure courante porte le numéro i.
Pour tracer un deuxième graphique sur la même figure, taper `hold on`. L'effet de cette commande s'annule avec `hold off`.

L'écriture de texte sur les axes d'un graphe ou sur le graphe s'effectue à l'aide des commandes :

`title`	titre du graphe
`xlabel`	titre de l'axe des abcisses
`ylabel`	titre de l'axe des ordonnées
`gtext`	placement d'un texte avec la souris
`text`	positionnement d'un texte spécifié par des coordonnées

La commande `grid` place une grille sur un graphe.

Les commandes précédentes doivent être saisies après la commande `plot`.

Par défaut les courbes sont tracées par des lignes continues, mais il y d'autres choix : `- -`, `:`, `- .`, `.`, `+`, `*`, `o` et `x.` De même, on peut préciser les couleurs avec : `y`, `m`, `c`, `r`, `g`, `b`, `w` et `k`

Par exemple, `plot(x,y,'r*')` va tracer en rouge et avec des * la courbe de y en fonction de x.

La commande `subplot` permet de partitionner un graphe en plusieurs graphes : `subplot(m, n, p)` divise la fenêtre de la figure en une matrice $m \times n$ de petits sous-graphes et sélectionne le $p^{ème}$ sous-graphe pour la figure courante.

Les commandes `semilogx` et `semilogy` à la place de `plot` permettent d'avoir respectivement l'axe des abcisses ou des ordonnées en échelle logarithmique.

5.7 Application 1 : probabilités et statistiques

Les commandes `normrnd ou randn` permettent de générer des valeurs suivant une loi gaussienne :

Exemple : `bruit=sqrt(sigma2)*randn(1,N);`

ou `bruit=normrnd(0,sqrt(sigma2),1,N);`

Remarque : *Il est possible de générer un bruit non gaussien (voir* `help stats` *pour d'autres générateurs de nombres aléatoires. Si on souhaite ajouter un bruit coloré il faudra d'abord générer un bruit blanc (avec* `randn` si il est gaussien*) et, ensuite, le filtrer en utilisant la fonction* ***`filter`***

Exemple *:* `y = filter(B,A,z);`

Si on utilise un filtre de type RIF on prendra `A=1`.

Question 1 : En utilisant la commande `randn`, générer 1000 réalisations indépendantes $x_1, ..., x_{1000}$, d'une variable aléatoire gaussienne centrée de variance 4. Visualiser l'histogramme et déterminer la moyenne et l'écart type de la séquence générée à l'aide des fonctions `hist`, `mean` et `std`.
Calculer la moyenne et l'écart type de la séquence $x_1^2, ..., x_{1000}^2$. Comparer avec les résultats théoriques.

Question 2 : A l'aide de la commande `rand`, générer 1000 réalisations indépendantes de la variable aléatoire discrète X définie par :

$$\begin{cases} P[X = 1] = 1/4 \\ P[X = 0] = 1/2 \\ P[X = -1] = 1/4 \end{cases}$$

Calculer la moyenne de la séquence générée.

5.8 Application 2 : traitement du signal

Question 1 : Représenter dans le plan polaire (commande `polar`) les pôles (commande `roots`) du filtre suivant :

$$\frac{1}{A(z)} = \frac{1}{1 - 1.45z^{-1} + 0.81z^{-2}}$$

Question 2 : Générer le signal :

$$x(t) = 10\sin(0.4\pi t + \phi) + b(t), \qquad t = 0, ..., 127$$

où ϕ est une variable aléatoire uniformément distribuée sur $[0, 2\pi)$ et $b(t)$ est un bruit blanc gaussien de variance unité. Etudier la moyenne, la fonction d'autocorrélation (fonction `xcorr`) et le spectre de $x(t)$ (commande `fft`). Comparer les résultats obtenus aux résultats théoriques.

Question 3 : En créant une fonction, générer un signal N.R.Z. soumis à une modulation d'amplitude (multiplication par $\cos(2\pi f_0 t)$). Etudier le spectre du signal avant et après modulation. Comparer les résultats obtenus aux résultats théoriques.

5.9 Conclusion

Ceci n'était qu'un aperçu des capacités de Matlab. Ce logiciel possède de nombreuses boîtes à outils liées au traitement du signal et aux probabilités élémentaires : `signal` (traitement du signal), `stats` (statitiques), `wavelets` (ondelettes), `nnet` (réseaux de neurones), `optim` (optimisation), `hosa` (moments d'ordres supérieurs)... Pour avoir un aperçu des possibilités de Matlab et de ces différentes boîtes à outils, taper `demo`.