TRAVAUX PRATIQUES DE TRAITEMENT DU SIGNAL (1)

1 • PRÉALABLES.

Cette séance de travaux pratiques a pour but de vous faire expérimenter les notion de stationnarité, d'ergodicité, de densité spectrale de puissance à travers une application simple d'identification de la réponse impulsionnelle d'un filtre.
Vous disposez, pour cette séance, d'un fichier de 800 réalisations d'un même signal de 1000 échantillons temporel et d'une fonction de filtrage. Le but est d'identifier la réponse impulsionnelle du filtre en utilisant des signaux aléatoires.
Il ne s'agit pas de travaux pratiques de génie informatique : ce qui est important dans cette séance c'est de bien interpréter les données, pas de les calculer.

2 • STATIONNARITÉ - ERGODICITÉ.

2.1 • Les signaux.

2.2 • Rappels sur les moments statistiques.

Si x est un signal aléatoire, alors chaque trajectoire de ce signal est différente. On nomme k $(k = 1, \ldots, K)$ le numéro de la trajectoire et t le temps. Donc, pour la trajectoire k et au temps t, le signal réel x prend la valeur $x_k(t)$. Nous nous intéressons ici à des signaux discrets, donc connus uniquement en N temps discrets $t_n = n\delta t$ $(n = 0, \ldots, N-1)$ et δt est la période d'échantillonnage.

-> • Moment d'ordre 1.

A chaque instant t_n, on peut calculer $m_x(t_n)$ la moyenne des réalisations du signal aléatoire par : $m_x(t_n) = \frac{1}{K}\sum_{k=1}^{K} x_k(t_n)$. La vraie valeur de $m_x(t_n)$ ne peut être obtenue qu'avec un nombre infini de trajectoires (i.e. $K \to \infty$), mais on peut se rapprocher de cette valeur si K est suffisamment grand.
Un signal est dit **stationnaire d'ordre 1** si $m_x(t_n)$ ne dépend pas du temps. On note alors $m_x(t_n) = m_x$.

-> • Moment d'ordre 2.

Pour chaque couples d'instants $(t_n, t_{n'})$ on peut calculer l'auto-covariance du signal x de la même façon : $M_{xx}(t_n, t_{n'}) = \frac{1}{K}\sum_{k=1}^{K} x_k(t_n)x_k(t_{n'})$.

Un signal est dit **stationnaire d'ordre 2** si $M_{xx}(t_n, t_{n'})$ ne dépend que de $(t_n - t_{n'}) = (n - n')\delta t$. On pose alors $\tau = (t_n - t_{n'})$ et on note

$M_{xx}(t_n, t_{n'}) = R_{xx}(t_n - t_{n'}) = R_{xx}(\tau)$. $R_{xx}(\tau)$ est une fonction discrète ($\tau_m = m\delta t$, ici m peut être négatif) qui peut alors être estimé par :

$R_{xx}(\tau_m) = \frac{1}{K}\sum_{k=1}^{K} x_k(t_n)x_k(t_n + \tau_m)$ en faisant bien attention que les signaux $x_k(t_n)$ $x_k(t_n + \tau_m)$ existent, ce qui veut dire qu'on ne peut pas calculer R_{xx} pour toute valeur de t_n. Dans la pratique, on estime l'auto-covariance que pour un nombre restreint de valeurs de τ : $\tau_m \in [-M\delta t, M\delta t]$ et on ne calcule R_{xx} que sur (N - 2M) valeurs de temps. On aura donc (N - 2M) calculs de R_{xx} (pour toutes valeurs de t_n avec $n \in [M, N - M - 1]$). On peut donc avoir une meilleure estimation de R_{xx} en calculant la moyenne de ces différentes réalisations de R_{xx}.

2.3 • Vérifiez que votre signal est bien stationnaire.

Les 800 réalisations du signal aléatoire sont disponibles dans le fichier `signal_aleatoire`. Pour charger les 800 réalisation du signal à étudier tapez :

```
>> load signal_aleatoire ;
```

Vérifiez que vous possédez bien une matrice de 800 signaux de 1000 échantillons sous le nom de `signal_entree`.

Comme on ne dispose que d'estimations de la moyenne et de la fonction d'auto-covariance, on ne peut pas facilement vérifier la stationnarité (il y a peu de chance que $m_x(t_n)$ ne fluctue pas). On va donc faire une simple vérification statistique en traçant l'histogramme des $m_x(t_n)$ pour les 1000 valeurs de t_n et on regarder la distribution de ces valeurs. Pour ce faire, vous pouvez en regarder l'évolution temporelle,

```
>> figure(32) ; plot(mx) ;
```

ou en en visualisant la distribution en utilisant (par exemple) la fonction `hist` de Matlab. Pour vérifier stationnarité d'ordre 1, il faut tracer un histogramme en une dimension, pour la stationnarité d'ordre 2, il faudrait tracer un histogramme à deux dimensions. La fonction `hist` ne fonctionnant que pour des données mono-dimensionnelles, vous vous contenterez de vérifier la stationnarité d'ordre 1. Le plus important ici est de bien interpréter l'histogramme que vous obtenez.

On supposera que le signal est stationnaire d'ordre 2. R_{xx} peut alors être estimé par la moyenne des différentes réalisations (c'est à dire les différentes valeurs de $R_{xx}(\tau)$ pour une valeur de t_n donnée. Tracez les différentes réalisations de R_{xx} et sa moyenne. Pour le calcul pratique, on prendra M=100, c'est à dire que $\tau_m \in [-100\delta t, 100\delta t]$.

Remarquez que les fluctuations de R_{xx} ne sont pas très importantes et que la forme générale est conservée quel que soit l'instant t_n considéré.

Dans la suite du travail, on prendra la moyenne des différentes réalisations de l'auto-covariance comme valeur pour R_{xx}.

2.4 • Rappels sur les moments temporels.

-> • Moment d'ordre 1.

Si les moments statistiques sont calculés pour t constant, les moments temporels peuvent être calculés pour chaque trajectoire. On note : $\mu_x(k) = \lim_{T \to \infty} \frac{1}{2T} \int_{-T}^{T} x_k(t)dt$. Dans la pratique, sur des signaux discrets, cette procédure revient à calculer la moyenne de tous les échantillons de x_k. $\mu_x(k) = \frac{1}{N} \sum_{n=1}^{N} x_k(t_n)$. Les différents $\mu_x(k)$ peuvent être considérés comme des réalisations de la variable μ_x. Le signal est dit **ergodique d'ordre 1** si $m_x = \mu_x$.

-> • Moment d'ordre 2.

Pour chaque valeur de décalage τ_m on peut calculer l'auto-corrélation du signal x par:

$$C_{xx}(\tau_m) = \frac{1}{N-m} \sum_{n=\max(1,\,1-m)}^{\min(N-m,\,N)} x_k(t_n)x_k(t_n+\tau_m).$$

On peut donc calculer une auto-corrélation par trajectoire. On considérera en final la moyenne de toutes ces auto-corrélations.

Un signal est dit **ergodique d'ordre 2** si $R_{xx}(\tau_m) = C_{xx}(\tau_m),\ \forall \tau_m$.

2.5 • Vérifiez que votre signal est bien ergodique.

On vérifie l'ergodicité à l'ordre 1 en traçant l'histogramme des μ_x et en le comparant à l'histogramme de m_x. L'intersection des deux histogrammes doit être forte. Dans la pratique, il faudrait réaliser un test statistique, mais cette procédure sort du programme du cours. On se contente donc de le vérifier qualitativement.

On considérera par la suite que ce signal est centré.

Calculez $C_{xx}(\tau_m)$ pour toutes les valeurs de $\tau_m \in [-M\delta t, M\delta t]$ (avec M=100). On simplifiera la formule précédente en :

$$C_{xx}(\tau_m) = \frac{1}{N-2M} \sum_{n=M+1}^{N-M} x_k(t_n)x_k(t_n+\tau_m),$$

qui est plus simple à programmer. Calculez la moyenne des C_{xx} calculés pour chaque trajectoire. Tracez le et comparez le qualitativement à R_{xx}.

2.6 • Densité spectrale de puissance et transformée de Fourier.

Pour chaque trajectoire de votre signal, calculez la transformée de Fourier de la fonction d'auto-corrélation ($\Gamma_{xx}(\omega)$, on suppose qu'on est ergodique d'ordre 2), d'une part, et la transformée de Fourier du signal lui-même, d'autre part. Affichez la séquence des modules (ou de la partie réelle) de ces transformées de Fourier. Que constatez-vous ? comment interprétez-vous ce résultat ?
Comme vous calculez la transformée de Fourier sur un échantillon réduit, il est préférable d'utiliser une fenêtre de Barlett dont l'expression est :

$F_B(\tau) = \left(1 - \frac{|\tau|}{M}\right) \mathbb{1}_{[-M, M]}(\tau)$, où $\mathbb{1}_{[-M, M]}(\tau) = 1$ si $\tau \in [-M, M]$, et 0 sinon.

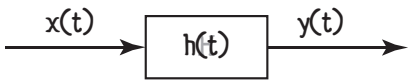
-> • Rappels Matlab.
Si `signal` est le signal dont vous devez calculer et afficher le module (ou la partie réelle) de la transformée de Fourier, la séquence d'ordres est :

```
>> fenetre = 1:length(signal) ; fenetre = fenetre - (length(signal)/2) ;
>> fenetre = ( 1 - (abs(fenetre)) / fenetre(length(fenetre)) ) ;
>> signal_fenetre = signal .* fenetre ;
>> TF_signal = fftshift(fft(signal_fenetre)) ;
>> figure(1) ; clf ; plot(abs(TF_signal)) ; % pour afficher le module
>> figure(1) ; clf ; plot(real(TF_signal)) ; % pour afficher la partie reelle
```

3 • Systèmes linéaires et signaux aléatoires.

3.1 • Rappels sur les relations de base.

Soit un système linéaire dont la réponse impulsionnelle est $h(t)$, dont la fonction de transfert est $H(\omega)$, la transformée de Fourier de $h(t)$. Soit $x(t)$ un signal aléatoire ergodique au sens large placé en entrée de ce système et $y(t)$ la sortie de ce système quand $x(t)$ en est l'entrée. L'auto-corrélation de $x(t)$ est $R_{xx}(\tau)$ dont la transformée de Fourier est $\Gamma_{xx}(\omega)$, l'auto-corrélation de $y(t)$ est $R_{yy}(\tau)$ dont la transformée de Fourier est $\Gamma_{yy}(\omega)$, l'inter-corrélation entre $y(t)$ et $x(t)$ est $R_{yx}(\tau)$ dont la transformée de Fourier est $\Gamma_{yx}(\omega)$. La moyenne du signal $x(t)$ (rsp. de $y(t)$) est m_x (rsp. m_y).



On a alors :

$m_y = H(0)m_x$ $\Gamma_{yx}(\omega) = H(\omega)\Gamma_{xx}(\omega)$ et $\Gamma_{yy}(\omega) = |H(\omega)|^2\Gamma_{xx}(\omega)$.

3.2 • Expérimentation.

Le filtre vous est fourni sous la forme d'une fonction compilée de Matlab. Si le signal à filtrer se nomme `signal_entrant` on peut obtenir le résultat de son filtrage sous le nom de `signal_sortant`. Il suffit pour ce faire d'écrire :

```
>> [signal_sortant] = Filtre_mystere(signal_entrant) ;
```

Le filtre ne fonctionne que pour un signal monodimentionnel. Vous ne pouvez pas filtrer la totalité de vos signaux avec cette fonction. Vous devez utiliser la fonction avec les 800 réalisations du signal $x(t)$ issu du fichier `signal_aleatoire` et récupérer 800 réalisations du signal de sortie :

```
>> N_expe = 800 ;
>> for k=1:N_expe
>>  signal_sortie(k,:) = Filtre_mystere(signal_entree(k,:)) ;
>> end
```

Essayez d'utiliser cette expérience pour visualiser le module de $H(\omega)$. Pour ca, il vous faudra calculer la covariance $R_{yx}(\tau)$ et sa transformée de Fourier.

Que pouvez-vous en déduire sur la nature du filtre en question ?

3.3 • Identification du filtre.

On va supposer que le filtre que vous devez caractériser peut être approximé par un filtre de réponse impulsionnelle finie sur un horizon de 10 échantillons dont la fonction de transfert (en z) s'écrit : $H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + \ldots + h_{10} z^{-10}$, c'est à dire que la relation (discrète) entrée/sortie est $y_k = h_0 x_k + h_1 x_{k-1} + h_2 x_{k-2} + \ldots + h_{10} x_{k-10}$.

Dans ce cas là, les 11 paramètres $(h_0, \ldots, h_{10})$ peuvent être identifiés par l'équation :

$$\begin{bmatrix} R_{xx}(0) & \ldots & R_{xx}(10) \\ \ldots & \ldots & \ldots \\ R_{xx}(-10) & \ldots & R_{xx}(0) \end{bmatrix} \begin{bmatrix} h_0 \\ \ldots \\ h_{10} \end{bmatrix} = \begin{bmatrix} R_{yx}(0) \\ \ldots \\ R_{yx}(-10) \end{bmatrix}.$$

Mettez en place cette résolution. Testez la différence entre les deux filtres avec un signal d'entrée composite dont l'expression est :

```
>> delta_t = 0.001 ;
>> temps = 0:delta_t:10 ;
>> amplitude = randn(1,10) ;
>> frequence = rand(1,10) * (0.03/(2*delta_t)) ;
>> signal_test = temps ;
>> signal_test(floor(length(signal_test)/2):length(signal_test)) =
signal_test(floor(length(signal_test)/2):length(signal_test)) + 20 ;
>> for k=1:10
>>  signal_test = signal_test + amplitude(k)*sin(2*pi*frequence(k)*temps) ;
>> end
>>
>> signal_test = signal_test + 0.2*randn(size(signal_test)) ;
>> signal_filtre = Filtre_mystere(signal_test) ;
>> signal_filtre_bis = 0*signal_test ;
>>
>> for k=length(h)+1:length(signal_test)
>>  for t=1:length(h)
>>   signal_filtre_bis(k) = signal_filtre_bis(k) + h(t)*signal_test(k-t+1) ;
>>  end
>> end
```

On a supposé ici que le vecteur des 11 valeurs non-nulles de votre réponse impulsionnelle finie est stockée dans un vecteur nommé `h`.

4 • Quelques conseils pour la programmation.

Déclarez toujours vos variables. Créez des vecteurs explicites pour toutes vos manipulations. Par exemple créez un vecteur τ :

```
>> tau_max = 100 ;
>> tau = -tau_max:tau_max ;
```

N'oubliez pas que les tableaux sous Matlab commencent à un, ce qui pose des problèmes surtout quand on doit stocker $R_{xx}(\tau)$ alors que τ peut avoir des valeurs négatives.

Une façon de contourner ce problème est de créer l'indice pour lequel τ vaut 0 :

```
>> indice_0 = tau_max + 1 ;
```

Alors on peut indexer un tableau `Rxx` par :

```
>> Rxx = zeros(1,length(tau)) ;
>> for k=1:length(Rxx)
>>  Rxx(indice_0 - tau(k)) = 1.0 ;
>> end
```

De façon explicite, `Rxx(indice_0 - tau(k))` correspond à $R_{xx}(\tau_k)$.