Université Djilali Bounaama de Khemis Miliana Faculté des Sciences et de la Technologie

Département de Technologie/ Niveau et Spécialité: 1ère Année Master Telecom

Matière : TP Signaux aléatoires et Processus stochastiques

TP N°03 : Densité spectrale de puissance (Analyse et Simulation d'un bruit blanc sous Matlab)

I) Rappels théoriques

a) Définition : Un processus aléatoire (ou un signal pour votre visualisation) avec une fonction de densité spectrale de puissance (PSD) constante est un processus de bruit blanc.

b) Densité spectrale de puissance

La fonction de densité spectrale de puissance (PSD) indique la quantité de puissance contenue dans chacune des composantes spectrales. Par exemple, pour une onde sinusoïdale de fréquence fixe, le tracé PSD ne contiendra qu'une seule composante spectrale présente à la fréquence donnée.

c) Bruit blanc gaussien et uniforme

Un bruit blanc (processus) est constitué d'un ensemble de variables aléatoires indépendantes et identiquement distribuées (i.i.d). Au sens discret, le bruit blanc constitue une suite d'échantillons indépendants et générés à partir de la même distribution de probabilité.

Par exemple, vous pouvez générer un bruit blanc à l'aide d'un générateur de nombres aléatoires dans lequel tous les échantillons suivent une distribution gaussienne donnée. C'est ce qu'on appelle le bruit blanc gaussien (WGN : peut-être généré à l'aide de la fonction *randn*). De même, un bruit blanc généré à partir d'une distribution uniforme est appelé bruit blanc uniforme (peut être généré à l'aide de la fonction *rand*).

Le bruit gaussien et le bruit uniforme sont fréquemment utilisés dans la modélisation de systèmes. Prenons l'exemple de la génération d'un bruit blanc gaussien de longueur 10 à l'aide de la fonction randn dans Matlab - avec une moyenne nulle et un écart type = 1.

```
%White Noise:Simulation and Analysis  using Matlab
 mu=0;sigma=1;  noise= sigma *randn(1,10)+mu
```

d) Bruit blanc strictement et faiblement défini

Étant donné que le processus de bruit blanc est construit à partir d'échantillons/variables aléatoires i.i.d, tous les échantillons suivent la même fonction de distribution de probabilité. Ainsi, la fonction de distribution de probabilité conjointe du processus ne changera pas avec un décalage dans le temps. C'est ce qu'on appelle un processus stationnaire. Un processus stationnaire peut être classé en processus **stationnaire au sens strict** (Strict Sense Stationary :SSS) ou **stationnaire au sens large(** Wide Sense Stationary :WSS). En conséquence, ils peuvent être appelés bruit blanc strictement défini(SSS) et bruit blanc faiblement défini(WSS).

e) Fonction et matrice de covariance : Un bruit blanc, noté (t) , est défini au sens faible comme une condition plus pratique. Ici, les échantillons sont statistiquement non corrélés et distribués de manière identique avec une variance égale à σ^2. Cette condition est spécifiée en utilisant une fonction de covariance comme

$$COV(x_i, x_j) = \begin{cases} \sigma^2 & ,i = j \\ 0 & ,i \neq j \end{cases}$$

Pourquoi avons-nous besoin d'une fonction de covariance ?

Car, nous avons affaire à un processus aléatoire composé de variables aléatoires (10 variables dans l'exemple de modélisation ci-dessus). Un tel processus est considéré comme un vecteur aléatoire ou une variable aléatoire multivariée. Dans ce cas, la fonction de covariance spécifie comment chacune des variables du processus aléatoire donné se comporte les unes par rapport aux autres. La fonction de covariance généralise la notion de variance à plusieurs dimensions. L'équation ci-dessus, lorsqu'elle est représentée sous forme matricielle, donne la matrice de covariance du processus aléatoire de bruit blanc. Étant donné que les variables aléatoires de ce processus ne sont pas corrélées statistiquement, la fonction de covariance ne contient des valeurs que le long de la diagonale.

$$C_{XX} = \begin{pmatrix} \sigma^2 & \cdots & 0 \\ \vdots & \sigma^2 & \vdots \\ 0 & \cdots & \sigma^2 \end{pmatrix} = \sigma^2 \mathbf{I}$$

La matrice ci-dessus indique que seule la fonction d'auto-corrélation existe pour chaque variable aléatoire. Les valeurs de corrélation croisée sont nulles (les échantillons/variables sont statistiquement non corrélés les uns par rapport aux autres). Les éléments diagonaux sont égaux à la variance et tous les autres éléments de la matrice sont nuls. La fonction d'auto-corrélation d'ensemble du bruit blanc faiblement défini est donnée par :

$$R_{XX}(\tau) = E[x(t)x^*(t-\tau)] = \sigma^2\delta(\tau)$$

Cela indique que la fonction d'auto-corrélation du processus de bruit blanc faiblement défini est nulle partout sauf au décalage $\tau = 0$.

Caractéristiques dans le domaine fréquentiel

Le théorème de Wiener-Khintchine énonce que pour le processus stationnaire au sens large (WSS), la fonction de densité spectrale de puissance $S_{xx}(f)$ d'un processus aléatoire peut être obtenue par la transformée de Fourier de la fonction d'auto-corrélation du processus aléatoire. Dans le domaine temporel continu, ceci est représenté par :

$$S_{XX}(f) = F[R_{XX}(\tau)] = \int_{-\infty}^{\infty} R_{XX}(\tau)e^{-j2\pi f\tau}\, d\tau$$

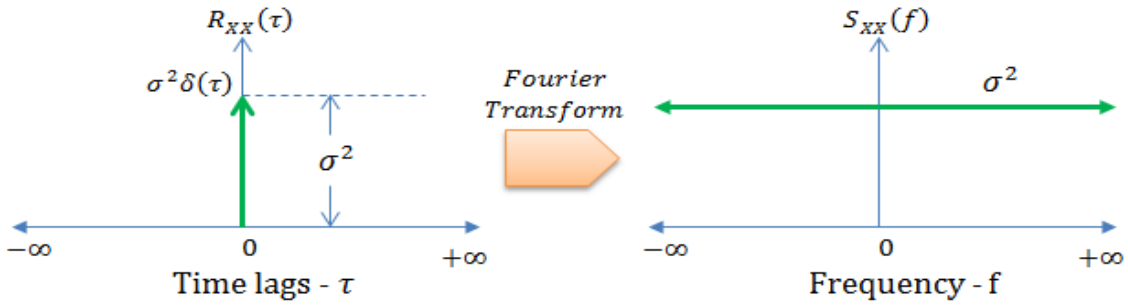


Figure 1: Illustration du Théorème de Weiner-Khintchine

Pour le processus de bruit blanc faiblement défini, nous constatons que la moyenne est une constante et que sa covariance ne varie pas par rapport au temps. C'est une condition suffisante pour un processus WSS. Ainsi, nous pouvons appliquer le théorème de Weiner-Khintchine. Par conséquent, la densité spectrale de puissance du processus de bruit blanc faiblement défini est constante (plate) sur l'ensemble du spectre de fréquences (Figure 1). La valeur de la constante est égale à la variance ou puissance du signal de bruit.

$$S_{XX}(f) = F[R_{XX}(\tau)] = \int_{-\infty}^{\infty} \sigma^2\delta(\tau)e^{-j2\pi f\tau}\, d\tau = \sigma^2 \int_{-\infty}^{\infty} \delta(\tau)e^{-j2\pi f\tau}\, d\tau = \sigma^2$$

II) Simulations des caractéristiques du bruit blanc gaussien dans Matlab

1) Générez un bruit blanc gaussien de longueur $L = 100,000$ à l'aide de la fonction *randn* dans Matlab et tracez-le. Supposons que la densité de probabilité est gaussienne avec une moyenne $\mu = 0$ et un écart type $\sigma = 2$. Ainsi, la variance est $\sigma^2 = 4$. La densité de probabilité théorique de la gaussienne est donnée par :

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

2) clear all; clc; close all;

```
L=100000; %Sample length for the random signal
mu=0; sigma=2;
X=sigma*randn(L,1)+mu;
figure();
subplot(2,1,1), plot(X);
title(['White noise : \mu_x=',num2str(mu),' \sigma^2=',num2str(sigma^2)])
xlabel('Samples')
ylabel('Sample Values')
grid on;
```

3) Tracez l'histogramme du bruit généré et comparez le à l'histogramme de la densité de probabilité théorique de la variable aléatoire gaussienne.

```
subplot(2,1,2)
n=100; %number of Histrogram bins
[f,x]=hist(X,n);
bar(x,f/trapz(x,f)); hold on;
%Theoretical PDF of Gaussian Random Variable
g=(1/(sqrt(2*pi)*sigma))*exp(-((x-mu).^2)/(2*sigma^2));
plot(x,g);hold off; grid on;
title('Theoretical PDF and Simulated Histogram of White Gaussian Noise');
legend('Histogram','Theoretical PDF');
xlabel('Bins');
ylabel('PDF f_x(x)');
```

4) Calculer la fonction d'auto-corrélation du bruit blanc.

La fonction d'auto-corrélation calculée doit être mise à l'échelle correctement. Si la fonction « xcorr » (intégrée à Matlab) est utilisée pour calculer la fonction d'auto-corrélation, utilisez l'argument « biaisé » dans la fonction pour la mettre à l'échelle correctement.

```
figure();
Rxx=1/L*conv(flipud(X),X);
lags=(-L+1):1:(L-1); plot(lags,Rxx);
title('Auto-correlation Function of white noise');
xlabel('Lags')
ylabel('Correlation')
grid on;
```

```
%Alternative method
%[Rxx,lags] =xcorr(X,'biased');
%The argument 'biased' is used for proper scaling by 1/L
%Normalize auto-correlation with sample length for proper scaling
```

5) **Simulation de la densité spectrale de puissance (PSD) :** Simuler la densité spectrale de puissance (PSD) du bruit blanc est une tâche un peu délicate. Il y a deux problèmes ici :

a) Les échantillons générés sont de longueur finie. Cela revient à appliquer la troncature d'une série infinie d'échantillons aléatoires. Cela implique que les décalages sont définis sur une plage fixe.

b) Les générateurs de nombres aléatoires utilisés dans les simulations sont des générateurs pseudo-aléatoires. Pour ces deux raisons, vous n'obtiendrez pas un spectre plat de PSD lorsque vous appliquez la transformée de Fourier sur les valeurs d'auto-corrélation générées. L'effet d'oscillation du PSD peut être minimisé en générant un signal aléatoire suffisamment long et en faisant la moyenne du PSD sur plusieurs réalisations du signal aléatoire.

4.1. Simulation du bruit blanc gaussien en tant que vecteur aléatoire gaussien multivarié :

Pour vérifier la densité spectrale de puissance du bruit blanc, nous utiliserons l'approche consistant à envisager le bruit comme la somme de N variables aléatoires gaussiennes. Nous voulons faire la moyenne de la DSP sur L de telles réalisations. Puisqu'il existe N variables aléatoires gaussiennes (N échantillons individuels) par réalisation, la matrice de covariance C_{xx} sera de dimension $N \times N$. Le vecteur de moyenne pour ce cas multivarié sera de dimension $1 \times N$.La décomposition de Cholesky de la matrice de covariance donne l'écart type équivalent pour le cas multivarié. La décomposition de Cholesky peut être considérée comme une opération de racine carrée. La fonction ***randn*** de Matlab est utilisée ici pour générer le processus aléatoire gaussien multidimensionnel avec la matrice moyenne et la matrice de covariance données.

```
mu=0; %Mean of each realization of Noise Process
sigma=2; %Sigma of each realization of Noise Process

L = 1000; %Number of Random Signal realizations to average
N = 1024; %Sample length for each realization set as power of 2 for FFT
 %Generating the Random Process - White Gaussian Noise process
MU=mu*ones(1,N); %Vector of mean for all realizations
Cxx=(sigma^2)*diag(ones(N,1)); %Covariance Matrix for the Random Process
R = chol(Cxx); %Cholesky of Covariance Matrix
%Generating a Multivariate Gaussian Distribution with given mean vector and
%Covariance Matrix Cxx
z = repmat(MU,L,1) + randn(L,N)*R;
```

Calculez la PSD du processus multidimensionnel généré ci-dessus et faites-en la moyenne pour obtenir un tracé lisse.

```
Z = 1/sqrt(N)*fft(z,[],2); %Scaling by sqrt(N);

Pzavg = mean(Z.*conj(Z));%Computing the mean power from fft
 normFreq=[-N/2:N/2-1]/N;
Pzavg=fftshift(Pzavg); %Shift zero-frequency component to center of spectrum
plot(normFreq,10*log10(Pzavg),'r'); title('power spectral density')
axis([-0.5 0.5 0 10]); grid on; ylabel('Power Spectral Density (dB/Hz)'); xlabel('Normalized Frequency');
```

Application : Dans la modélisation de canal, nous rencontrons souvent un canal de bruit blanc gaussien additif (AWGN).

III) Le périodogramme

On génère un signal composé d'une onde sinusoïdale de 100 Hz accompagné d'un bruit additif N(0,1). La fréquence d'échantillonnage est de 1 kHz. La longueur du signal est de 1000 échantillons.

```
fs = 1000;  t = 0:1/fs:1-1/fs;
x = cos(2*pi*100*t) + randn(size(t));
figure,plot(x)
```

a) Périodogramme en utilisant la FFT.

Le signal est à valeur réelle et a une longueur paire. Étant donné que le signal est à valeur réelle, vous n'avez besoin que d'estimations de puissance pour les fréquences positives ou négatives. Afin de conserver la puissance totale, multipliez toutes les fréquences qui se produisent dans les deux ensembles (les fréquences positives et négatives) par un facteur de 2. La fréquence zéro (DC) et la fréquence de Nyquist ne se produisent pas deux fois. Tracez le résultat.

```
N = length(x);
xdft = fft(x);
xdft = xdft(1:N/2+1);
psdx = (1/(fs*N)) * abs(xdft).^2;
psdx(2:end-1) = 2*psdx(2:end-1);
freq = 0:fs/length(x):fs/2;

plot(freq,pow2db(psdx))
grid on
title('Periodogram Using FFT')
xlabel('Frequency (Hz)')
ylabel('Power/Frequency (dB/Hz)')
```

b) **Périodogramme à l'aide de l'instruction « périodogram ».**

```
figure
periodogram(x,rectwin(N),N,fs)
mxerr = max(psdx'-periodogram(x,rectwin(N),N,fs))
```

Montrer que les deux résultats sont identiques.

c) Entrée à fréquence normalisée

```
%%c)    Entrée à fréquence normalisée
N = 1000;
n = 0:N-1;
x = cos(pi/4*n) + randn(size(n));
xdft = fft(x);
xdft = xdft(1:N/2+1);
psdx = (1/(2*pi*N)) * abs(xdft).^2;
psdx(2:end-1) = 2*psdx(2:end-1);
freq = 0:2*pi/N:pi;
figure
plot(freq/pi,pow2db(psdx))
grid on
title('Periodogram Using FFT')
xlabel('Normalized Frequency (\times\pi rad/sample)')
ylabel('Power/Frequency (dB/(rad/sample))')
figure
periodogram(x,rectwin(N),N)
mxerr = max(psdx'-periodogram(x,rectwin(N),N))
```

Identification des systèmes (estimation de la densité spectrale de puissance

I. Objectifs du TP

1. Analyser les estimateurs de la densité spectrale de puissance
2. Tdentifier des systemes en utilisant le procédé de correlation

II. Partie théorique

Le périodogramme permet une estimation simple de la densité spectrale de puissance en prenant le carré de la transformée de Fourier.

TP 02 : Identification des Systèmes

(Estimation de la Densité Spectrale de Puissance)

I. Objectifs du TP

1. Analyser les estimateurs de la densité spectrale de puissance (DSP).
2. Identifier des systèmes en utilisant le procédé de corrélation.

II. Partie théorique

- **Le périodogramme** permet une estimation simple de la densité spectrale de puissance en prenant le carré de la transformée de Fourier. Il a été introduit par Arthur Schuster en 1898. L'estimation de la d.s.p du signal x est :

$$\hat{S}_{per}(\omega) = \frac{1}{N}\left|\sum_{n=1}^{N} x(n)exp(-j\omega n)\right|^2$$

Autrement dit :

$$\hat{S}_{per}(f) = \frac{|fft(x)|^2}{N}$$

N représente le nombre d'échantillons fixés. ω représente la pulsation ($w = 2\pi f$).

- **Périodogramme moyenné**

$$\hat{S}_{per}(f) = \frac{|fft(x)|^2}{N}$$

N représente le nombre d'échantillons fixés. ω représente la pulsation ($w = 2\pi f$).

- **Périodogramme moyenné**

Pour améliorer les performances de l'estimateur précédent on calcule plusieurs périodogrammes sur des signaux indépendants. La sinusoïde est la même pour les différents signaux mais le bruit est indépendant pour chaque réalisation. On calcule ensuite la moyenne des différents périodogrammes. Cette méthode réduit la variance d'un facteur égal au nombre de périodogrammes calculés.

- **Fonction d'auto-corrélation (FAC)** :

À un processus stochastique discret ou continu, correspond une « auto-corrélation » statistique qui généralise la notion de covariance. Dans le cas d'un processus continu (en toute généralité complexe) X(t), la fonction d'auto-corrélation statistique se définit comme :

$$R_X(t_1, t_2) = E[X(t_1).X^*(t_2)]$$

Avec : $t_1 = t \quad t_2 = t - \tau$:

$$\boldsymbol{R_{XX}(\tau) = E[X(t_1)X^*(t_2)] = E[X(t)X^*(t-\tau)]}$$

La d.s.p de X, $S_{XX}(f)$ est la transformée de Fourier (TF) de la FAC

- **Périodogramme Lissé (de Welch)**

La méthode de Welch fournit un estimateur consistant de la densité spectrale de puissance. Cette méthode a été proposée par Peter D. Welch en 1967

La méthode de Welch consiste à calculer plusieurs périodogrammes à partir d'un unique signal en utilisant une fenêtre glissante. Il s'agit d'une fenêtre rectangulaire de taille très inférieure à la taille du signal glissant d'échantillon en échantillon. Cette méthode réduit le biais de l'estimateur. Les différentes fenêtres sont : Hamming, Hannin, Bartlett

- **Caractéristiques des Estimateurs**

Chacun de ces estimateurs peut être caractérisé par une étude statistique dont les résultats sont ***le biais*** et ***la variance***.

Manipulations

Manipulation1

Génération du signal bruité et estimation de la dsp par le periodogramme

Dans cette partie on s'intéresse aux méthodes du « periodogram », periodogramme moyenné et périodogramme lissé ou« welch » pour l'estimation de la DSP.

Pour réaliser cette manipulation, on procéde de la manière suivante :

1. Génération d'un signal sinusoïdal de fréquence 100 Hz, et de durée 1.5s.

On se pose que la fréquence d'échantillonnage est de 1000 Hz.

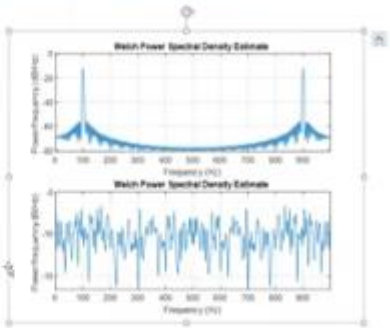
2. La génération du bruit en utilisant la fonction **'awgn'** (le cas ou SNR= -20).
3. L'utilisation de la fonction 'periodogram' pour calculer et tracer la DSP par la methode du periodogramme.

```
clc;
clear all;
close all;
f=100;fe=1000 ;t=0:1/fe:1.5;
s=cos(2*pi*f*t);
sb=awgn(s,-20);
figure
subplot 211
plot(s);
subplot 212
plot(sb);
```

```
% La méthode du periodogram
figure
subplot 211
periodogram(s,[],[],fe,'twosided');
subplot 212
periodogram(sb,[],[],fe,'twosided');
```

Manipulation2 : periodogramme fenétré (periodogram de Welch : pwelch

```
% La méthode de welch
figure
subplot 211
pwelch(s,[],[],[],fe,'twosided');
subplot 212
pwelch(sb,[],[],[],fe,'twosided');
```



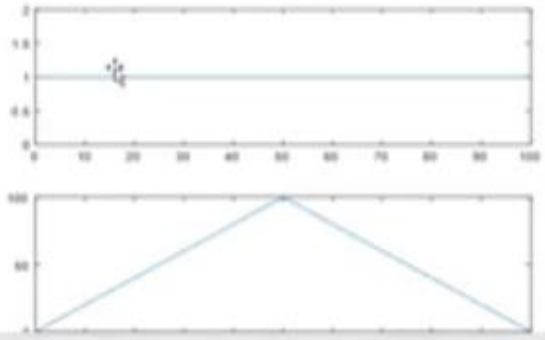
Identification des systèmes par corrélation

Dans cette partie du TP on calcule la fonction de corrélation des signaux non-périodiques (Corrélation linéaire) et des signaux périodiques (Corrélation circulaire).

A. Corrélation linéaire :

Génération d'un signal échelant (de 100 échantillons).
Utilisation de la fonction *'xcorr'* pour calculer la fonction d'autocorrélation.

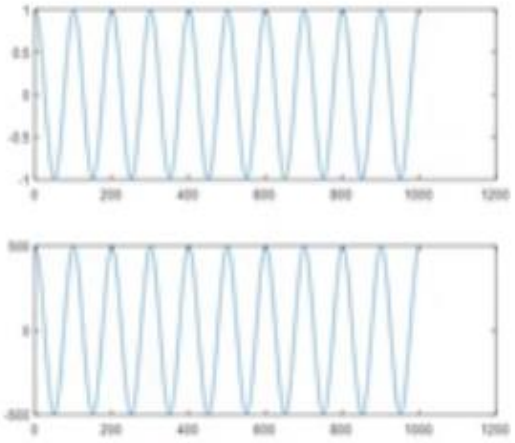
```
% Corrélation lineaire
x=ones(1,100);
rx=xcorr(x);
figure(1)
subplot 211
plot(x)
subplot 212
```



B Corrélation circulaire :

Génération d'un signal sinusoidal de fréquence 10 Hz,et le temp de pas 0.001. (de 100 échantillons).
Utilisation de la transformée de la DSP avec la fonction *'fft'* et *'ifft'*, *'xcorr'* pour calculer la fonction d'autocorrélation.

```
% Corrélation circulaire
t=0:0.001:1;
xl=cos(2*pi*10*t);
rxl=fftshift(real(ifft((fft(xl)).*conj(fft(xl)))));
figure(2)
subplot 211
plot(xl)
subplot 212
plot(rxl)
```



Travail à effectuer

- Comparer les résultats de la manipulation 1 avec les résultats de cette fonction qu'il *faut créer sous matlab* :

```
function P = periodogramme(S)
N = length(S);
Y = fft(S,N);
Y = abs(Y);
P = (Y.^2)/N;
```

- Utiliser la fonction periodogramme précédent pour calculer et tracer le perodogramme moyéné:

```
Function P=periodo_moy(s, var_bruit)
for i=1 : nb_periodo
N=length(s) ;
bruit = randn(1,N)*var_bruit;
signal = s + bruit;
P1 = periodogramme(signal);
plot(20*log(P1(1:round(N/2))), 'b')
```

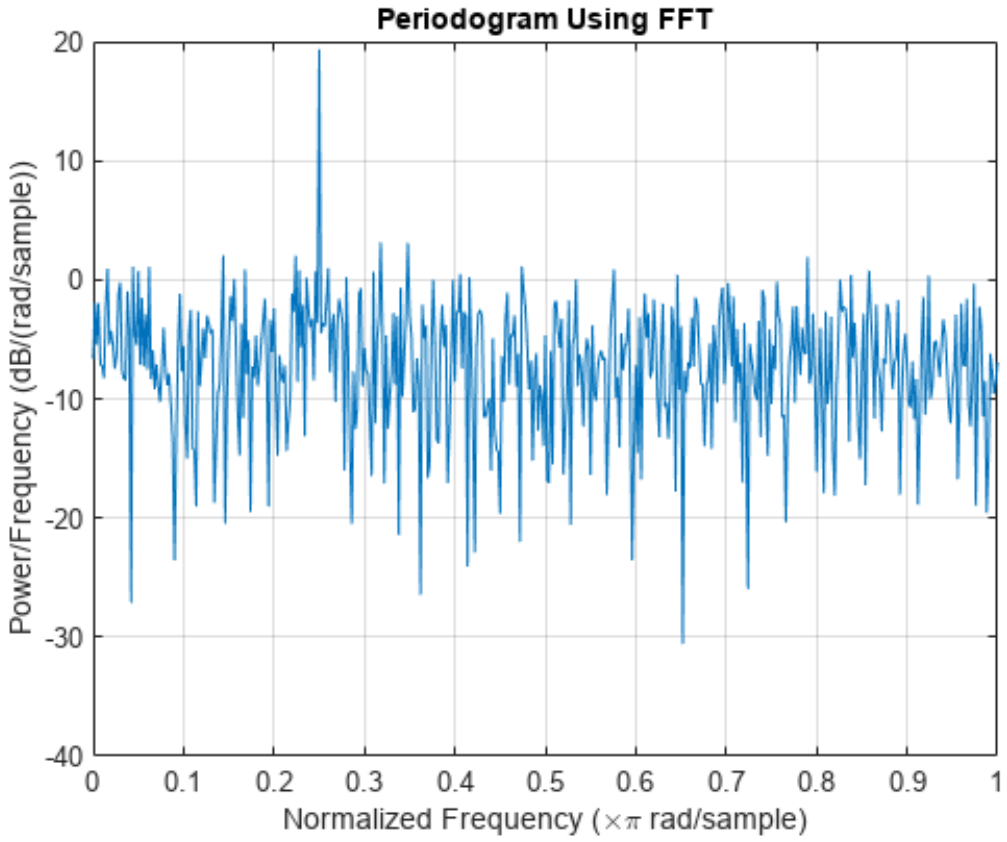
perodogramme moyéné:

```
Function P=periodo_moy(s, var_bruit)
for i=1 : nb_periodo
N=length(s) ;
bruit = randn(1,N)*var_bruit;
signal = s + bruit;
P1 = periodogramme(signal);
plot(20*log(P1(1:round(N/2))), 'b')
P = [P ; P1];
end
Pmoy = mean(P);
```

- Calculer les caractéristiques statistiques de chaque estimateur (**biais et variance**) par la fonction suivante :

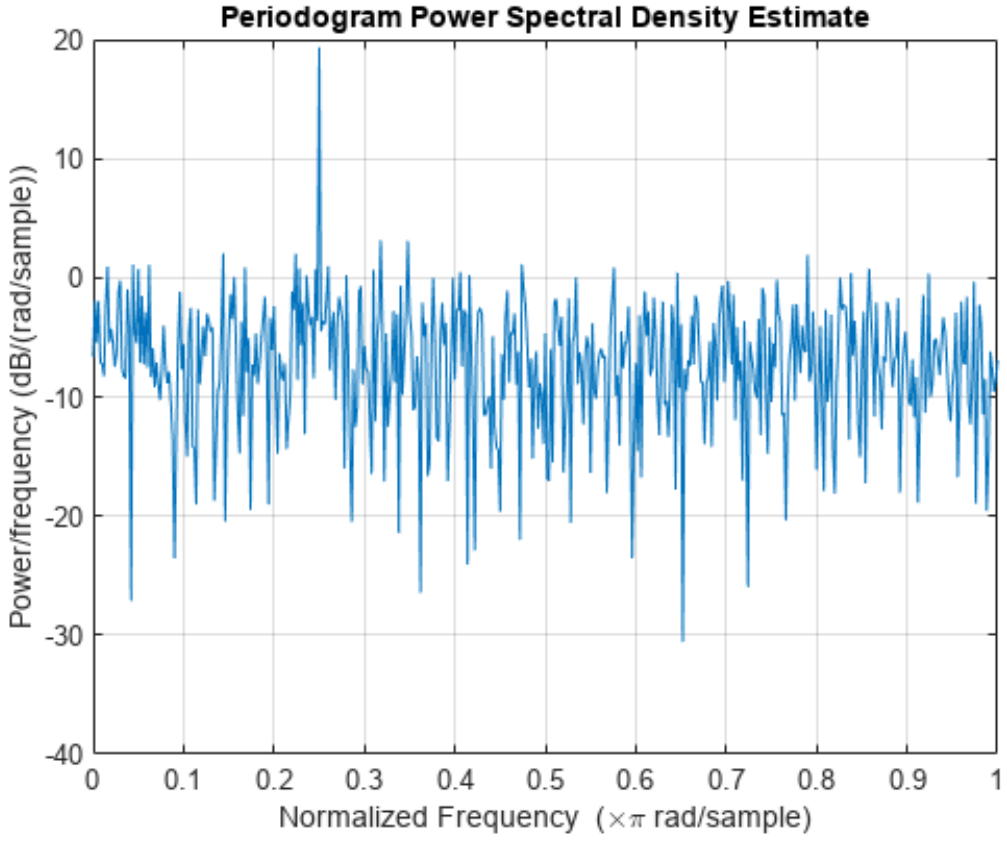
```
function [biais,variance]=stats(P)
N = length(P);
biais = mean(S);
variance = 0;
for i = 1 : N
 variance = variance + ((P(i)-biais)^2);
end
variance = variance / N;
```

Dans la méthode d'estimation de la dsp par Welch, on peut utiliser plusieurs fenètres , à savoir : Hamming, Hanning, Bartlett. Faites varier les fenêtres pour estimer et tracer la dsp par Welch et calculer à chaque fois les caractéristiques statistiques par la fonction (stats). (pour cela utiliser pwelch et préciser la fenêtre)



Compute and plot the periodogram using periodogram. Show that the two results are identical.

periodogram(x,rectwin(N),N)



mxerr = max(psdx'-periodogram(x,rectwin(N),N))

mxerr = 4.4409e-16

Complex-Valued Input with Normalized Frequency

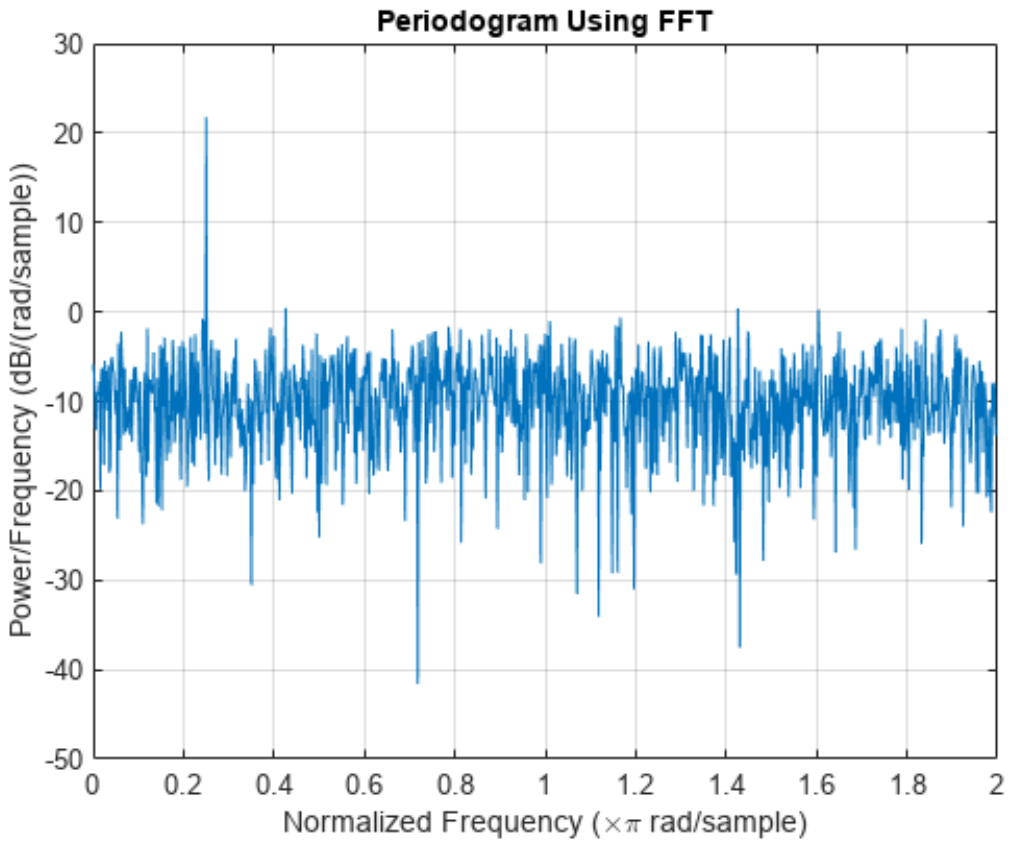
Use fft to produce a periodogram for a complex-valued input with normalized frequency. The signal is a complex exponential with an angular frequency of $\pi/4$ rad/sample in complex-valued $N(0,1)$ noise.

```
N = 1000;
n = 0:N-1;
x = exp(1j*pi/4*n) + [1 1j]*randn(2,N)/sqrt(2);
```

Use fft to obtain the periodogram. Because the input is complex-valued, obtain the periodogram from $[0,2\pi)$ rad/sample. Plot the result.

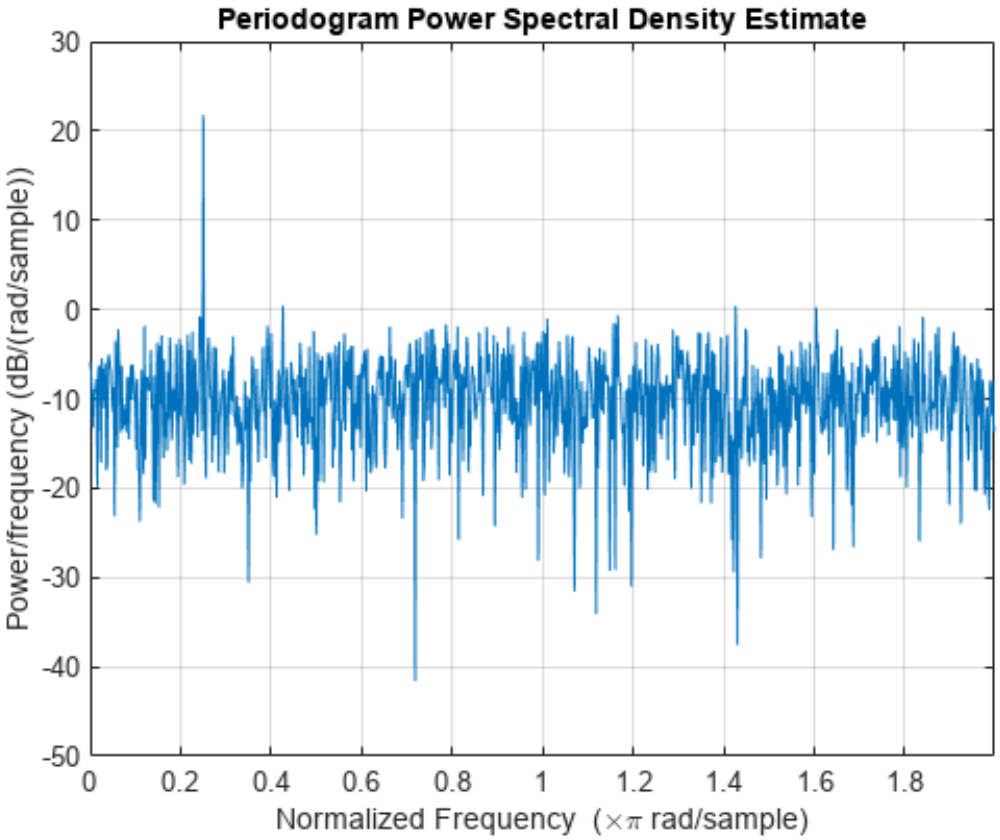
```
xdft = fft(x);
psdx = (1/(2*pi*N)) * abs(xdft).^2;
freq = 0:2*pi/N:2*pi-2*pi/N;

plot(freq/pi,pow2db(psdx))
grid on
title("Periodogram Using FFT")
xlabel("Normalized Frequency (\times\pi rad/sample)")
ylabel("Power/Frequency (dB/(rad/sample))")
```



Use periodogram to obtain and plot the periodogram. Compare the PSD estimates.

```
periodogram(x,rectwin(N),N,"twosided")
```




```
mxerr = max(psdx'-periodogram(x,rectwin(N),N,"twosided"))
```