Université des Sciences et Technologies de Lille
U.F.R. de Mathématiques Pures et Appliquées
Bât. M2, F-59655 Villeneuve d'Ascq Cedex

Agrégation externe **Année 2003-2004**

Corrigé de l'exercice test de Neyman-Pearson et détection radar

Commençons par quelques rappels du cours. On considère un modèle statistique $\big(\Omega, \mathcal{F}, (P_\theta)_{\theta\in\Theta}\big)$. En pratique, on prend pour P_θ la loi de l'échantillon observé (considéré comme vecteur aléatoire à composantes i.i.d.) lorsque la valeur du paramètre inconnu est θ. On se donne deux parties disjointes Θ_0 et Θ_1 de Θ. Le problème est de décider après observation d'un $\omega \in \Omega$, en pratique après observation d'un échantillon numérique $(x_1, \ldots, x_n)$, si la vraie valeur du paramètre est dans Θ_0 ou dans Θ_1.

On privilégie l'une des deux hypothèses : $(\mathcal{H}_0) : \theta \in \Theta_0$, appelée hypothèse nulle (celle à laquelle on a « envie » de croire). L'autre hypothèse $(\mathcal{H}_1) : \theta \in \Theta_1$ est appelée *hypothèse alternative*. Il importe de noter que Θ_1 n'est pas forcément égal à tout le complémentaire de Θ_0 dans Θ.

Une statistique de test φ est une application mesurable $\Omega \to [0,1]$, donc en pratique une fonction mesurable du vecteur échantillon observé. Le test associé à φ est défini par la règle de décision suivante :

- Si $\varphi(\omega) = 0$, on accepte $(\mathcal{H}_0)$.
- Si $\varphi(\omega) = 1$, on rejette $(\mathcal{H}_0)$.
- Si $\varphi(\omega) = p \in]0,1[$, on rejette $(\mathcal{H}_0)$ avec probabilité p et on l'accepte avec probabilité $1-p$.

On dit que le test est *déterministe* si φ ne peut prendre que les valeurs 0 et 1. Sinon, on parle de test *randomisé* ou *stochastique*. Dans le cas d'un test déterministe, on définit la région critique ou région de rejet $R := \varphi^{-1}(\{1\}) = \{\omega \in \Omega;\ \varphi(\omega) = 1\}$. Clairement dans ce cas, la fonction φ peut s'écrire $\varphi = \mathbf{1}_R$.

La décision prise à l'issue du test peut donner lieu à deux types d'erreurs

- *l'erreur de première espèce* qui consiste à rejeter $(\mathcal{H}_0)$ alors qu'elle est vraie,
- *l'erreur de deuxième espèce* qui consiste à accepter $(\mathcal{H}_0)$ alors qu'elle est fausse.

	$(\mathcal{H}_0)$ vraie	$(\mathcal{H}_0)$ fausse
Acceptation de $(\mathcal{H}_0)$	Pas d'erreur	Erreur deuxième espèce
Rejet de $(\mathcal{H}_0)$	Erreur première espèce	Pas d'erreur

On contrôle l'erreur de première espèce par le *niveau* α_φ du test, défini par

$$\alpha_\varphi := \sup_{\theta\in\Theta_0} \mathrm{E}_\theta\, \varphi. \tag{1}$$

L'erreur de deuxième espèce est contrôlée par la quantité

$$\sup_{\theta\in\Theta_1} \big(1 - \mathrm{E}_\theta\, \varphi\big). \tag{2}$$

On définit aussi la fonction *puissance* du test par

$$\beta_\varphi : \Theta_0 \cup \Theta_1 \to [0,1], \quad \theta \mapsto \mathrm{E}_\theta\, \varphi. \tag{3}$$

Dans ces formules, $\mathrm{E}_\theta\, \varphi$ désigne l'espérance de la variable aléatoire φ sous la loi P_θ. Il est utile de noter que $\mathrm{E}_\theta\, \varphi$ est la probabilité de rejeter $(\mathcal{H}_0)$ quand la vraie valeur du paramètre est θ (cf. cours). Dans le cas de deux hypothèses simples (*i.e.* $\Theta_0 = \{\theta_0\}$ et $\Theta_1 = \{\theta_1\}$), α est la probabilité de rejeter $(\mathcal{H}_0)$ alors qu'elle est vraie et $1 - \mathrm{E}_{\theta_1}\, \varphi$ est la probabilité d'accepter $(\mathcal{H}_0)$ alors qu'elle est fausse.

On dit que le test est *sans biais* si pour tout $\theta \in \Theta_1$, $\alpha_\varphi \leq \mathrm{E}_\theta\, \varphi$, autrement dit si la probabilité de rejeter $(\mathcal{H}_0)$ lorsqu'elle est vraie (*i. e.* lorsque $\theta \in \Theta_0$) est toujours inférieure ou égale à la probabilité de rejeter $(\mathcal{H}_0)$ lorsqu'elle est fausse (*i. e.* lorsque $\theta \in \Theta_1$).

Ex 1. *Test de Neyman-Pearson et détection radar* [2]

Un radar actif de surveillance aérienne a des caractéristiques telles qu'une éventuelle cible réfléchit $N = 20$ impulsions lors d'un balayage. À l'aide d'un traitement adapté, ces N impulsions réfléchies en cas de présence de la cible fournissent un vecteur d'observations $(z_i)_{1 \leq i \leq N}$ avec

$$\begin{array}{ll} (\mathcal{H}_1) & z_i = A + b_i \quad \text{en présence de cible,} \\ (\mathcal{H}_0) & z_i = 0 + b_i \quad \text{en l'absence de cible,} \end{array}$$

où les b_i sont des variables aléatoires gaussiennes $\mathfrak{N}(0, \sigma)$ indépendantes modélisant les divers bruits (σ est connu). La figure 1 représente un échantillon d'observations sous $(\mathcal{H}_0)$ et sous $(\mathcal{H}_1)$ pour le même bruit. Voici le script ayant servi à la générer.

```
// affichage graphique bruit et signal+bruit

N=20;
sgm=0.6;rand('normal');Br=sgm.*rand(N,1);// bruit simulé
t=(1:N)';
A=0.8; SgBr=A+Br;// signal + bruit
xbasc();
xsetech([0,0,1,0.5]);
plot2d3(t,Br,style=2,rect=[0,-2,22,3],strf='175',leg='bruit')
xsetech([0,0.5,1,0.5]);
plot2d3(t,SgBr,style=2,rect=[0,-2,22,3],strf='175',leg='signal + bruit')
```

Dans ce contexte, les erreurs de première et de deuxième espèce ont une signification bien concrète :

	$(\mathcal{H}_0)$ vraie	$(\mathcal{H}_0)$ fausse
Acceptation de $(\mathcal{H}_0)$	Pas d'erreur	Non détection
Rejet de $(\mathcal{H}_0)$	Fausse alarme	Pas d'erreur

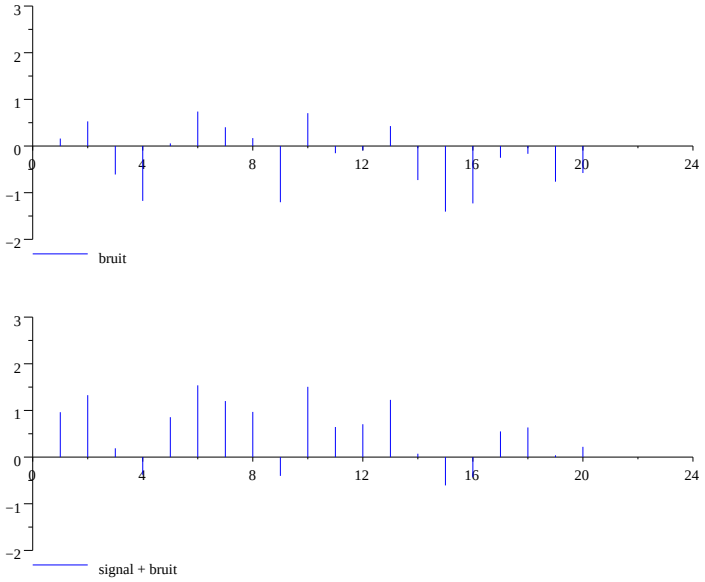

FIG. 1 – Bruit et signal bruité $A = 0.8$, $\sigma = 0.6$

1) On est exactement dans la situation donnée comme exemple en cours : test de l'hypothèse nulle « l'espérance vaut m_0 » contre l'hypothèse alternative « l'espérance vaut m_1 », avec m_0 et $m_1 > m_0$ connus, l'échantillon étant gaussien de variance connue σ^2. Avec les notations de l'énoncé, les variables aléatoires gaussiennes observées sont les z_i, $m_0 = 0$ et $m_1 = A$. On sait alors que le test de Neyman-Pearson de niveau α de $(\mathcal{H}_0)$ contre $(\mathcal{H}_1)$ a une région de rejet R de la forme

$$R = \left\{(z_1, \ldots, z_N) \in \mathbb{R}^N;\ \overline{z} > \frac{t_\alpha \sigma}{\sqrt{N}}\right\},$$

où $\overline{z} := (z_1 + \cdots + z_N)/N$ est la moyenne empirique calculée sur les observations z_i et t_α est le $1-\alpha$ quantile de la loi gaussienne standard $\mathfrak{N}(0,1)$, défini comme l'unique solution de l'équation

$$1 - \Phi(t) = \alpha, \tag{4}$$

Φ étant la f.d.r. de $\mathfrak{N}(0,1)$. On remarque que la région de rejet ne dépend aucunement de A. Voici un bout de script permettant de la calculer. La seule (petite) difficulté est l'utilisation de `cdfnor`.

```
alpha=10^(-6);
talpha=cdfnor("X",0,1,1-alpha,alpha);
N=20; sgm=0.6;
bornrej=talpha.*sgm.*N^(-0.5) // rejet si moyenne depasse cette valeur
```

En exécutant ce script, on obtient pour borne de rejet 0.6378. Voici maintenant comment vectoriser ce bout de script si on veut essayer diverses valeurs de α (copie d'écran, erreur sur `cdfnor` comprise...)

```
-->alpha=[0.05, 10.^(-(1:6))]
 alpha  =

!   0.05    0.1    0.01    0.001    0.0001    0.00001    0.000001 !

-->talpha=cdfnor("X",0,1,1-alpha,alpha);
cdfnor Mean,Std,P and Q must have same size
                                        !--error   999
talpha=cdfnor("X",0,1,1-alpha,alpha);

-->talpha=cdfnor("X",zeros(alpha),ones(alpha),1-alpha,alpha);

-->bornrej=talpha.*sgm.*N^(-0.5)
 bornrej  =

          column 1 to 5

!   0.2206803    0.1719382    0.3121123    0.4145982    0.4989584 !

         column 6 to 7

!   0.5721951    0.6377388 !
```

2) On se propose maintenant de simuler n échantillons de taille N sous l'hypothèse nulle, de leur appliquer le test de Neyman-Pearson et de calculer la fréquence empirique des *fausses alarmes*. D'après la loi des grands nombres et la définition du niveau α, qui s'interprète ici comme la probabilité de fausse alarme, cette fréquence empirique doit être voisine de α. On fera ensuite la même chose sous l'hypothèse alternative, pour évaluer la fréquence empirique de *non détection* des cibles réellement existantes. Par la loi des grands nombres, cette fréquence doit être voisine de la probabilité de non détection. Il est facile de calculer cette probabilité en remarquant que sous $(\mathcal{H}_1)$, $\overline{z}$ suit la loi gaussienne d'espérance A et d'écart type $\sigma N^{-1/2}$:

$$P_{\mathcal{H}_1}\left\{\overline{z} \le \frac{t_\alpha \sigma}{\sqrt{N}}\right\} = P_{\mathcal{H}_1}\left\{\frac{\sqrt{N}}{\sigma}(\overline{z} - A) \le t_\alpha - \frac{A\sqrt{N}}{\sigma}\right\} = \Phi\Big(t_\alpha - \frac{A}{\sigma}\sqrt{N}\Big). \qquad (5)$$

Un peu de réflexion montre qu'il est inutile de simuler deux matrices $N \times n$ d'observations gaussiennes. Il suffit d'en générer une sous l'hypothèse nulle et de s'en servir comme « bruit » sous $(\mathcal{H}_1)$, ce qui en pratique, revient à ajouter la constante A à chacun de ses termes. Ainsi pour 10 000 échantillons de taille 20, on aura économisé la simulation de 200 000 variables gaussiennes. Cette première économie n'est pas la seule possible, mais nous nous y tiendrons momentanément pour ne pas dévoyer la discussion.

Voici un script possible (`radarNP1.sce`).

```
// Test de Neyman Pearson et detection radar
```

```
printf('%s','Test de Neyman-Pearson et detection radar')
// Saisie et calcul des paramètres
alpha=0.001;
talpha=cdfnor("X",0,1,1-alpha,alpha);
N=20; sgm=0.6;
bornrej=talpha.*sgm.*N^(-0.5); // rejet si moyenne depasse cette valeur
ch1='Au niveau alpha = ';
ch2=', on rejette H0 si la moyenne dépasse ';
printf('%s%g%s%f',ch1,alpha,ch2,bornrej);
// simulation du bruit
n=10000; // nombre d'échantillons de taille N générés
printf('%s%d%s%d%s','On simule ',n,' échantillons de taille ',N,' du bruit')
rand("normal");
Z=sgm.*rand(N,n); // chaque colonne est un échantillon de N(0,sgm)
Zbar=sum(Z,'r')./N; // vecteur ligne des moyennes empiriques
// frequence empirique de fausses détections
FDfreq=sum(bool2s(Zbar>bornrej))./n;
printf('%s%.4f','Sous H0, fréquence empirique de fausses détections : ',FDfreq)
// %.4f impose 4 chiffres après la virgule pour fréquences
//
// Sous H1
A=0.7; // valeur du signal
printf('%s%g', 'Sous H1, avec valeur du signal A = ',A)
// Proba de non détection et puissance
[Pnd,Puiss]=cdfnor("PQ",talpha-(A./sgm).*N^(0.5),0,1);
printf('%s%g','  - probabilité de non détection d''une cible : ',Pnd)
// Observations sous H1
ZZ=A.*ones(Z)+Z; // avec signal, on recycle le meme bruit Z
ZZbar=sum(ZZ,'r')./N;//vecteur ligne des moyennes empiriques sous H1,
//
// frequence empirique des cibles non détectées
CNDfreq=sum(bool2s(ZZbar<=bornrej))./n;
printf('%s%.4f','  - fréquence empirique des cibles non détectées : ',CNDfreq);
```

Voici un exemple d'exécution.

```
-->;exec("/home/suquet/Enseignement/Scilab/Tests/radarNP1.sce");
Test de Neyman-Pearson et detection radar
Au niveau alpha = 0.001, on rejette H0 si la moyenne dépasse 0.414598
On simule 10000 échantillons de taille 20 du bruit
Sous H0, fréquence empirique de fausses détections : 0.0015
Sous H1, avec valeur du signal A = 0.7
  - probabilité de non détection d'une cible : 0.0166993
  - fréquence empirique des cibles non détectées : 0.0186
```

Regardons maintenant quelles économies on peut faire dans le script `radarNP1.sce` exposé ci-dessus. Comme le bruit est le même[1] sous $(\mathcal{H}_0)$ et sous $(\mathcal{H}_1)$, il est clair que $\overline{z}$ sous $(\mathcal{H}_1)$ est simplement égal à $\overline{z}$ sous $(\mathcal{H}_0)$ augmenté de A. En écrivant directement `ZZbar=A.*ones(Zbar)+Zbar ;`, on économise 190 000 additions et 10 000 divisions. Du coup, la matrice `ZZ` ne sert plus à rien et sa suppression nous économise la place mémoire occupée par 200 000 nombres réels ainsi que les 200 000 additions qui ont servi à sa création. Tant qu'à être radins, soyons le jusqu'au bout. Il est inutile de calculer `ZZbar`, il suffit de remplacer `bool2s(ZZbar<=bornrej)` par `bool2s(Zbar<=bornrej-A)`. Sous cette forme le gain est seulement celui de l'espace mémoire occupé par la variable `ZZbar` (donc 10 000 réels), par contre les 10 000 additions utilisées pour construire `ZZbar` ne sont peut-être pas économisées si `bornrej-A` est recalculé pour chacune des 10 000 composantes de `Zbar` (j'ignore ce qu'il en est réellement). Pour éviter ce gaspillage, on définit la variable `bornrejmA=bornrej-A` et on adopte finalement le code suivant qui remplace toute la fin du script à partir de « Observations sous H_1 » :

```
// Version économique sans simulation des observations sous H1
bornrejmA=bornrej-A;
CNDfreq=sum(bool2s(Zbar<=bornrejmA))./n;
printf('%s%.4f','  - fréquence empirique des cibles non détectées : ',CNDfreq);
```

3) En utilisant le script `radarNP1.sce` on peut se faire une première idée de l'influence des paramètres α, σ et A sur le comportement du test. Puisque le test est construit en fixant la probabilité α de fausse alarme, il est naturel de considérer la probabilité de non détection d'une cible existante comme une mesure de l'efficacité du test. Le test sera d'autant plus efficace que cette dernière probabilité sera plus petite, ou de manière équivalente que la probabilité complémentaire p (de détection d'une cible existante) sera plus proche de 1. D'après (5), on a

$$p = 1 - \Phi\Big(t_\alpha - \frac{A}{\sigma}\sqrt{N}\Big). \tag{6}$$

On peut alors faire les commentaires suivants.

1. Les paramètres A, σ et N étant fixés, p est une fonction croissante du niveau α. En effet Φ est croissante et t_α est une fonction décroissante de α, cf. (4). Ceci correspond bien à l'intuition : on a d'autant plus de chances de détecter une vraie cible que l'on est plus laxiste avec les fausses alarmes. Le cas limite consistant à prendre $\alpha = 1$, ainsi $t_\alpha = -\infty$ et $p = 1$. C'est bien entendu absurde en pratique puisque cela revient à décider qu'il y a une cible quelle que soit l'observation !
2. Les paramètres A et σ n'interviennent dans (6) que par leur rapport que l'on appelera avec un léger abus[2] *rapport signal sur bruit*. Clairement p est une fonction croissante de ce rapport, ce qui est là aussi conforme à l'intuition : la probabilité de détecter une vraie cible est d'autant plus élevée que le signal réfléchi par la cible est fort par rapport au bruit environnant.

[1] Attention, si on avait resimulé 200 000 gaussiennes pour passer à $(\mathcal{H}_1)$, ce raisonnement serait faux.
[2] En théorie du signal, le rapport signal sur bruit est plutôt A^2/σ^2.

3. Pour α et A/σ fixés, p est une fonction croissante de N et tend vers 1 quand N tend vers l'infini, ce qui ne surprendra personne !

Pour illustrer graphiquement l'influence de α et du rapport A/σ, on a représenté (figure 2) la courbe de p en fonction de α pour diverses valeurs de A/σ avec $N = 20$.

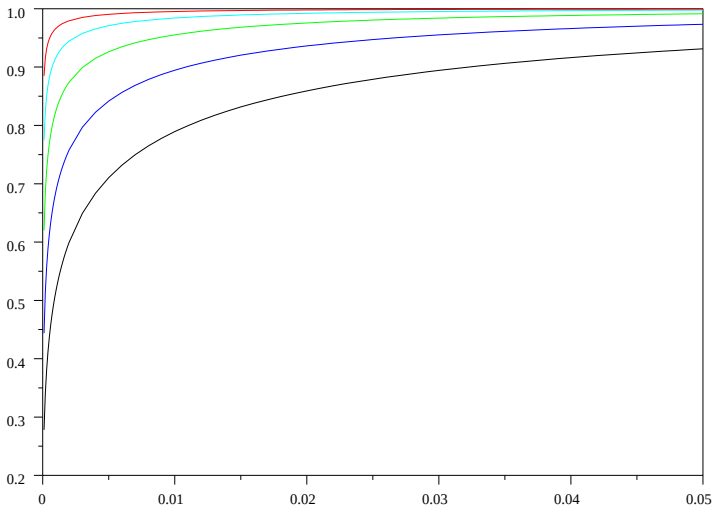

FIG. 2 – p en fonction de α pour $A/\sigma = 0.7, 0.8, 0.9, 1, 1.1$

Cette figure a été obtenue via le code Scilab suivant. On construit d'abord la fonction `Puissrad` :

```
function y=Puissrad(alpha,r,N)
// retourne la puissance du test radar Neyman-Pearson
// alpha est le niveau, r le rapport A/sigma et N la taille de l'échantillon
talpha=cdfnor("X",zeros(alpha),ones(alpha),1-alpha,alpha);
[Pnd,Puiss]=cdfnor("PQ",talpha-r.*N^(0.5),zeros(alpha),ones(alpha));
y=Puiss
endfunction
```

Ensuite,

```
-->;getf("/home/suquet/Enseignement/Scilab/Tests/Puissrad.sci");

-->alpha=[0.0001:0.0001:0.002, 0.002:0.001:0.05]';

-->p07=Puissrad(alpha,0.7,20);p08=Puissrad(alpha,0.8,20);

-->p09=Puissrad(alpha,0.9,20);p10=Puissrad(alpha,1,20);
```

```
-->p11=Puissrad(alpha,1.1,20);

-->p=[p07, p08, p09, p10, p11];

-->xbasc();plot2d(alpha,p)
```

Références

[1] D. DACUNHA-CASTELLE et M. DUFLO, *Probabilités et statistiques, 1. Problèmes à temps fixe*. Masson 1982.

[2] J.-P. DELMAS *Probabilités et télécommunications*. Exercices et problèmes commentés. Masson 1987.

[3] P. JAFFARD, *Statistique*, Résumé de cours-Exercices-Problèmes. Masson 1990.

[4] X. MILHAUD, *Statistique*. Editions espaces 34, Belin 2001.