Corrigé du TP1

1) Si on n'utilise pas une fonction déjà toute faite comme rnorm de Stixbox ou grand(n,'nor',mean,sd), on peut, par exemple, utiliser la méthode suivante, basée sur le fait qu'en coordonnées polaires, une loi normale $\mathcal{N}(O, I_2)$ suit la loi de densité $r\exp(-r^2/2)dr\frac{d\theta}{2\pi}$, donc ses marginales sont indépendantes et suivent pour R, la loi de fonction de survie $\mathbb{P}[R \geqslant r] = e^{-r^2/2}$, c'est–à–dire $\mathbb{P}[e^{-R^2/2} \leqslant u] = u$ et pour Θ , la loi uniforme sur $[0, 2\pi]$.

```
//              Simulation d'une loi normale N(0,I_2)
//              par la méthode de Box-Muller
//--------------------------------------------------------------------
// Entrées:
// x un vecteur dont seule la taille n compte
// Sorties:
// n1 une suite de n réalisations indépendantes de N(0,1)
// n2 idem et indépendantes de n1
function [n1,n2]=normale(x)
  theta = 2*%pi*grand(x,'def')
  rho   = sqrt(-2*log(grand(x,'def')))
  n1    = rho.*cos(theta)
  n2    = rho.*sin(theta)
endfunction
```

2) On peut par exemple utiliser la fonction suivante (qui n'est pas très bonne dans le sens où elle fait appel à la fonction rand() $2n$ fois pour finalement n'en utiliser que n sorties, mais elle s'écrit facilement à partir de la fonction précédente).

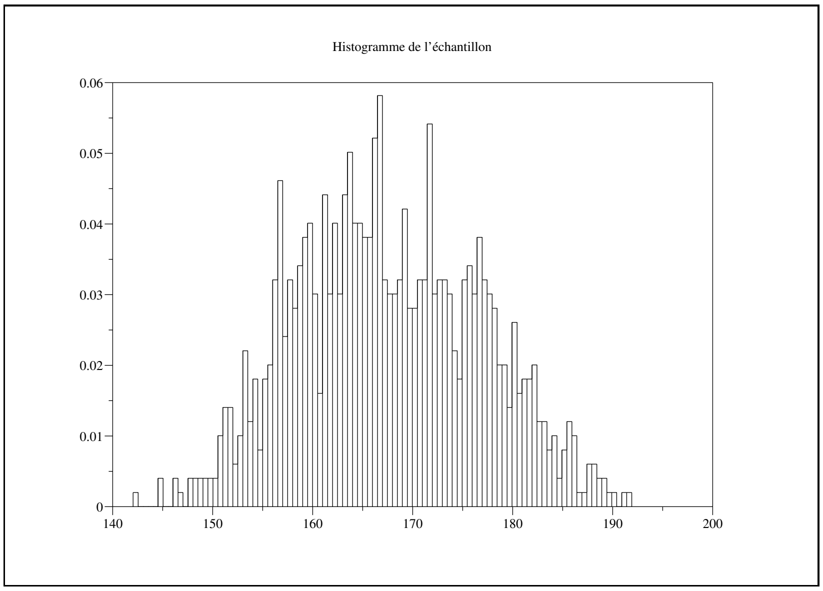
```
//      Simulation d'un mélange de deux lois normales univariées
//--------------------------------------------------------------------
// Entrées
// x un vecteur dont seule la taille n importe
// alpha le paramètre de mélange entre 0 et 1
// m1 la moyenne de la première gaussienne
// sigma1 l'écart-type de la première gaussienne
// m2 la moyenne de la seconde gaussienne
// sigma2 l'écart-type de la seconde gaussienne
// Sorties:
```

```
// ech une suite de n réalisations indépendantes du mélange
function ech=echantillon(x,alpha,m1,sigma1,m2,sigma2)
  melange     = (grand(x,'def')<alpha)
  [ech1,ech2] = normale(x)
  ech1        = m1 + (sigma1*ech1)
  ech2        = m2 + (sigma2*ech2)
  ech         = (ech1.*melange)+(ech2.*(~melange))
endfunction
```

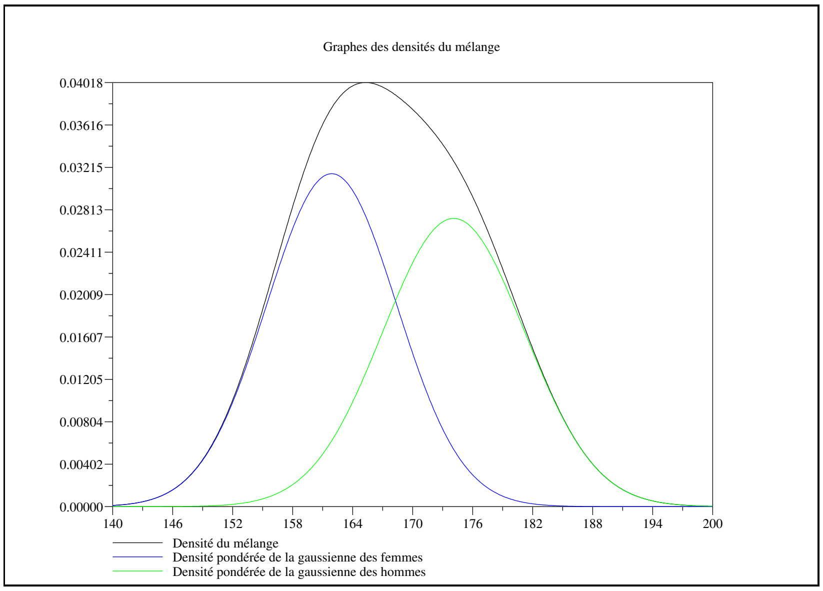
Le code suivant

```
// Réglage des paramètres
m1     = 161.9;
sigma1 = 6.5;
m2     = 174.1;
sigma2 = 7.1;

alpha  = 0.514;
n      = 1:1000;

// Simulation du mélange et histogramme de l'échantillon
ech = echantillon(n,alpha,m1,sigma1,m2,sigma2);
xset("window",0); xbasc();
histplot(100,ech)
xtitle("Histogramme de l''échantillon")

// Tracés des densités du mélange et des deux gaussiennes pures
t=140:0.1:200;
y=dmelange(t,alpha,m1,sigma1,m2,sigma2);
y1=alpha*dgaussienne(t,m1,sigma1);
y2=(1-alpha)*dgaussienne(t,m2,sigma2);

xset("window",1); xbasc();
plot2d(t',[y' y1' y2'],1:3,"121",..
"Densité du mélange@..
Densité pondérée de la gaussienne des femmes@..
Densité pondérée de la gaussienne des hommes")
xtitle("Graphes des densités du mélange")
```

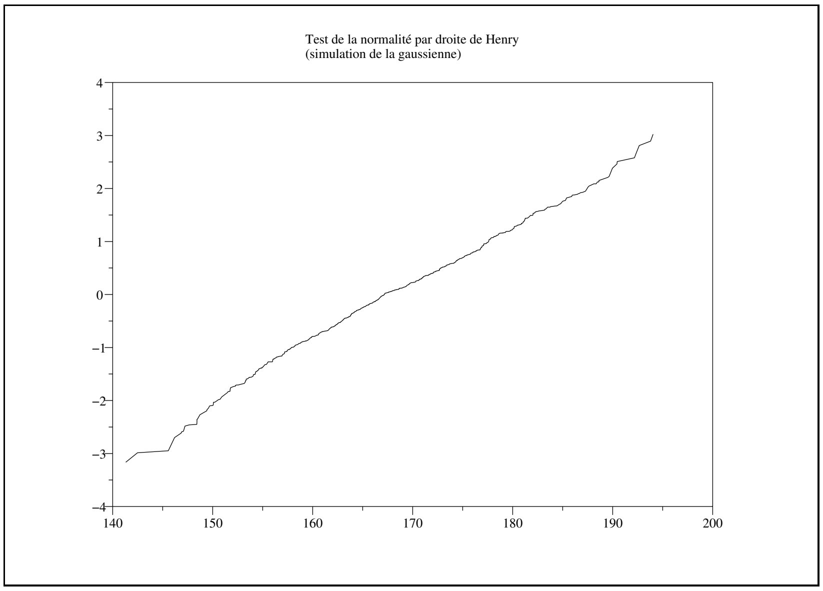
donne alors les graphiques



et



3) On peut utiliser pour tester l'adéquation de l'histogramme précédent à une loi gaussienne la méthode graphique “quantiles–quantiles” qui donne le graphe

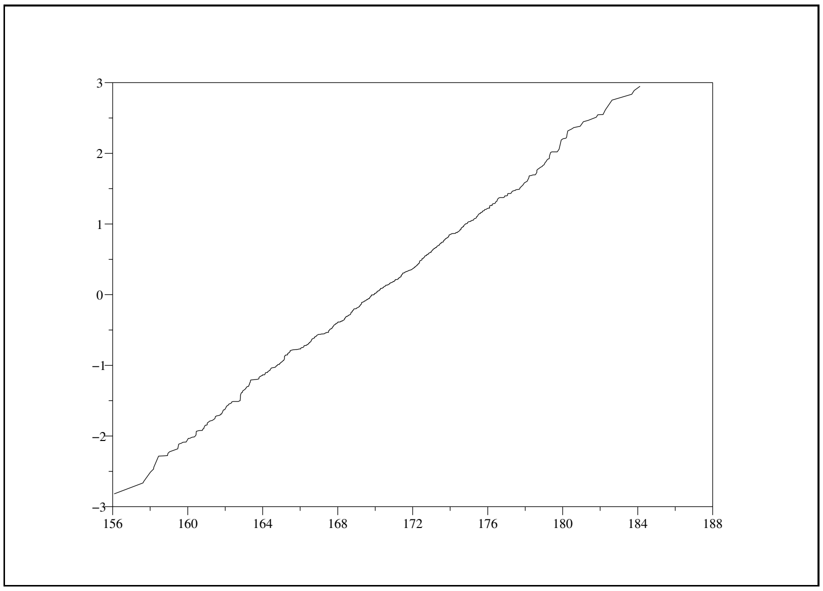


Pour produire ce graphique j'ai utilisé la fonction suivante

```
//           Droite de Henry approchée par simulation
//------------------------------------------------------------------
// Entrées
// ech L'échantillon à tester
// Sorties
// un graphe
function qqnorm_approx(ech)
  plot2d(sort(ech),sort(grand(ech,'nor',0,1)))
endfunction
```

En fait, il est plus précis d'utiliser la fonction de répartition inverse de la gaussienne, c'est ce que fait la fonction qqnorm de la bibliothèque Stixbox

Le résultat de ce test "à l'oeil nu" n'est pas clair dans ce cas, on a un semblant de droite. Comparons avec ce que donnerait le même test appliqué à un échantillon vraiment tiré suivant une loi normale (de moyenne 170 et d'écart-type 5.0)



Ceci est du au fait que l'échantillon est tiré suivant un mélange de deux gaussiennes de centre et variances assez proches.

Il est plus prudent d'effectuer un test quantitatif. Nous utiliserons le test de Shapiro-Wilk, avec le code suivant (logiciel R)

```
echantillon <- function(n,alpha,p1,p2) {
     k <- rbinom(1,size=n,prob=alpha);
  ech1 <- rnorm(k,mean=p1[1],sd=p1[2]);
  ech2 <- rnorm(n-k,mean=p2[1],sd=p2[2]);
  c(ech1,ech2)
}
pfem  <- c(161.9,6.5)
phom  <- c(174.1,7.1)
```

```
alpha <- 0.514
n     <- 1000
ech   <- echantillon(n,alpha,pfem,phom)

s <- shapiro.test(ech)
s
```

qui donne une p-value de $0,06$. On accepte "tout juste" le test si on se donne une tolérance de 5% : l'échantillon suit une loi normale d'après ce test. On effectue alors un test de Kolmogorov-Smirnov (question 7) entre l'échantillon de départ simulé suivant la loi du mélange et un échantillon de même taille tiré suivant la gaussienne estimée (ici par maximum de vraisemblance). Voici le code R :

```
ech2 <- rnorm(ech,mean=mean(ech),sd=sd(ech))
ks.test(ech,ech2)
```

et la sortie

```
Two-sample Kolmogorov-Smirnov test

data:  ech and ech2
D = 0.04, p-value = 0.4005
alternative hypothesis: two.sided
```

L'hypothèse est cette fois ci largement acceptée à 5%. Comment expliquer ce phénomène ? Les deux gaussiennes de distribution des tailles par sexe sont très proche. Il faut donc un échantillon de grande taille pour pouvoir discrimer entre le modèle de mélange et le modèle gaussien. En prenant un échantillon de taille 10 000 au lieu de 1 000, on obtient une p-value beaucoup plus petite pour le test KS.

4) Puisqu'on est capable de donner la vraisemblance du modèle, on peut faire une estimation par maximum de vraisemblance, qui, on le sait, présente des bonnes propriétés, comme celle d'exister dans le cadre du modèle exponentiel (ce qui n'est pas le cas ici malheureusement) et celle d'être *asymptotiquement efficace* (cf. [1]). Malheureusement, dans notre cas le maximum de la (log)-vraisemblance est égale à $+\infty$ (il est atteint pour une gaussienne dégénérée centrée sur un des points de l'échantillon. Ceci étant, on peut montrer sous des hypothèses assez générales (cf. [3]) que le point où la dérivée de la log-vraisemblance s'annule est unique et qu'il constitue un maximum local. C'est ce point qu'on va chercher.

5) Vu les spécifications du modèle, on a, en posant $\theta = (\alpha, m_F, \sigma_F, m_H, \sigma_H)$, et

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

et

$$f_\theta = \alpha f_{m_F,\sigma_F} + (1-\alpha) f_{m_H,\sigma_H}$$

$$\log \mathcal{L}_c^{(\theta)} = \sum_{i=1}^n \mathbb{1}_{z_i=0} \log[\alpha f_{m_F,\sigma_F}(x_i)] + \mathbb{1}_{z_i=1} \log[(1-\alpha) f_{m_H,\sigma_H}(x_i)]$$

Et aussi $\mathbb{E}_\theta[Z_1|X_1,\ldots,X_n] = \mathbb{E}_\theta[Z_1|X_1]$ par indépendance, puis

$$p_i \stackrel{\text{def}}{=} \mathbb{E}_\theta[Z_1|X_1] = \frac{\alpha f_{m_F,\sigma_F}(x_i)}{f_\theta(x_i)}$$

et

$$\begin{aligned} Q(\theta,\theta') &= \mathbb{E}_{\theta'}[\log \mathcal{L}_c^{(\theta)}|X_1,\ldots,X_n] = \mathbb{E}_{\theta'}\left[\mathbb{E}_{\theta'}[\log \mathcal{L}_c^{(\theta)}|Z_1,\ldots,Z_n]|X_1,\ldots,X_n\right] \\ &= \sum_{i=1}^n p_i' \log \alpha f_{m_F,\sigma_F}(x_i) + (1-p_i') \log(1-\alpha) f_{m_H,\sigma_H}(x_i) \end{aligned}$$

où les p_i' sont calculés pour la loi θ' et $\alpha, m_F, \sigma_F, m_H, \sigma_H$ pour la loi θ. On en déduit les estimateurs de maximum de vraisemblance classiques pour le modèle gaussien (ici pondérés par les p_i').

$$\hat{\alpha} = \frac{\sum_i p_i}{n}$$

$$\begin{aligned} \hat{m_F} &= \frac{\sum_i p_i' x_i}{\sum_i p_i'} & \hat{m_H} &= \frac{\sum_i (1-p_i') x_i}{\sum_i (1-p_i')} \\ \hat{\sigma_F^2} &= \frac{\sum_i (1-p_i')(x_i - m_F)^2}{\sum_i (1-p_i')} & \hat{\sigma_H^2} &= \frac{\sum_i (1-p_i')(x_i - m_H)^2}{\sum_i (1-p_i')} \end{aligned}$$

6) Des calculs qui précèdent, on déduit la fonction Scilab suivante qui code une itération de l'algorithme EM.

```
//                Une itération de l'algorithme EM
//-------------------------------------------------------------------
// Entrées
// ech l'échantillon
// alpha le paramètre de mélange entre 0 et 1
// m1 la moyenne de la première gaussienne
// sigma1 l'écart-type de la première gaussienne
// m2 la moyenne de la seconde gaussienne
// sigma2 l'écart-type de la seconde gaussienne
// Sorties
// logv la log vraisemblance pour les nouveaux paramètres
// a,... les nouveaux paramètres estimés
function [logv,a,mean1,sd1,mean2,sd2]=EMstep(ech,alpha,m1,sigma1,m2,sigma2)
  p=alpha*dgaussienne(ech,m1,sigma1)./dmelange(ech,alpha,m1,sigma1,m2,sigma2)
  a     = sum(p)
  mean1 = sum(p.*ech)/a
  sd1   = sqrt(sum(p.*(ech-mean1).*(ech-mean1))/a)
```

```
  mean2 = sum((1-p).*ech)/(length(p)-a)
  sd2   = sqrt(sum((1-p).*(ech-mean2).*(ech-mean2))/(length(p)-a))
  a     = a/length(p)
  logv  = sum(log(dmelange(ech,alpha,m1,sigma1,m2,sigma2)))
endfunction
```

En fait la version suivante est une légère modification de l'algorithme présenté dans l'énoncé. Voyez-vous de quelle modification il s'agit ?

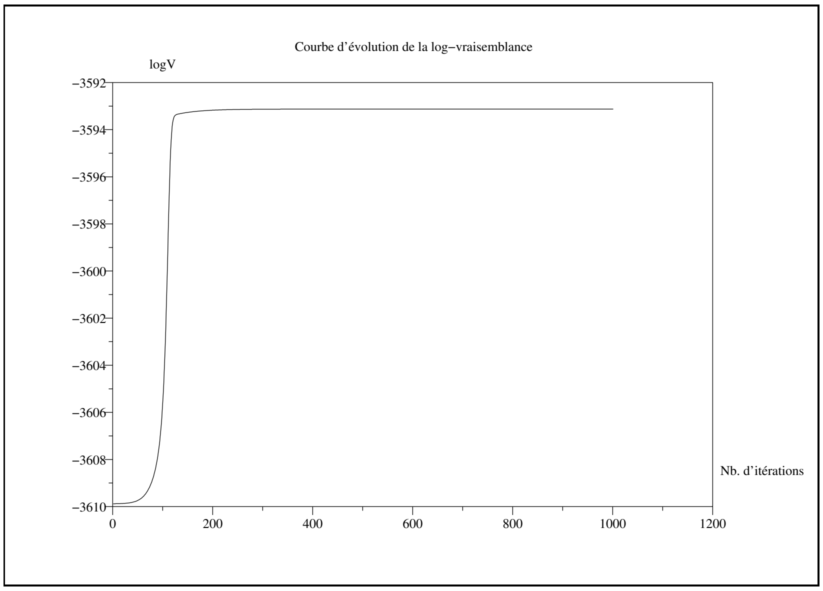
```
//       L'algorithme EM pour le mélange de deux lois normales
//-------------------------------------------------------------------
// Entrées
// ech l'échantillon
// pinit un vecteur de même taille que l'échantillon donnant
// l'inititalisation pour les paramètres cachés
// Sorties
// logv l'historique de l'évolution de la log-vraisemblance
// a,... les paramètres du mélange estimés par EM
function [logv,a,mean1,sd1,mean2,sd2]=EM(ech,pinit)
  a     = sum(pinit)
  mean1 = sum(pinit.*ech)/a
  sd1   = sqrt(sum(pinit.*(ech-mean1).*(ech-mean1))/a)
  mean2 = sum((1-pinit).*ech)/(length(pinit)-a)
  sd2   = sqrt(sum((1-pinit).*(ech-mean2).*(ech-mean2))/(length(pinit)-a))
  a     = a/length(pinit)
  iter  = %T
  //On fait une itération d'abord sinon la condition d'arrêt est réalisée
  //dès la première iteration de la boucle
  [lv,a,mean1,sd1,mean2,sd2] = EMstep(ech,a,mean1,sd1,mean2,sd2)
  logv  = []
  while iter,
    logv  = [logv lv]
    lvold = lv
    [lv,a,mean1,sd1,mean2,sd2]=EMstep(ech,a,mean1,sd1,mean2,sd2)
    if (lv-lvold <= 0) then
      iter = %F
    end
  end
endfunction
```

Les paramètres estimés sont dans mon cas :

$$\begin{aligned} \alpha &= 0.40 \\ m_F &= 163.30m \\ \sigma_F &= 7.11m \\ m_H &= 175.01m \\ \sigma_H &= 6.98m \end{aligned}$$

qui ne sont pas très loin des paramètres réels de la simulation, mis à part le α qui n'est pas terrible.

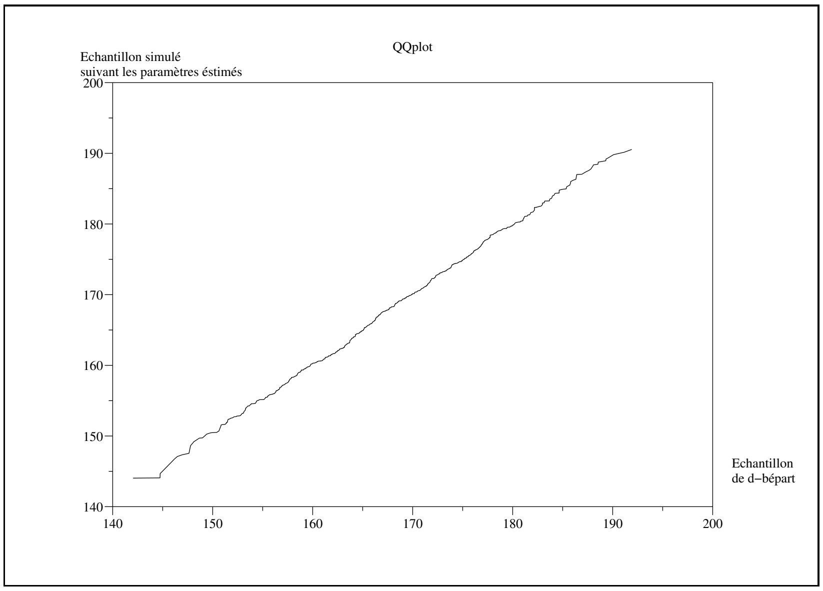
La courbe d'évolution de la log-vraisemblance est présentée ci-dessous :



Il a fallu de l'ordre de 1000 itérations à l'algorithme pour se figer complètement mais on constate qu'au bout des 100 premières itérations l'essentiel de l'optimisation était faite, ce sont les dernières décimales qui ont coûté le plus cher en temps de calcul (ici, la règle d'arrêt consistait à ce que la différence entre deux log-vraisemblances successives soit inférieure à la précision machine). On voit sur cet exemple qu'on a intérêt à fixer une règle d'arrêt moins exigeante.

7) On peut utiliser un graphe Quantile-Quantile entre les deux échantillon pour un test qualitatif.

```
ech_est = echantillon(ech,ALPHA,M1,SIGMA1,M2,SIGMA2);
plot2d(sort(ech),sort(ech_est))
```



Si on veut un test quantitatif, on peut utiliser le test de Kolmogorov-Smirnov implémenté par la fonction kstwo de la Stixbox :

```
[d,p]=kstwo(ech',ech_est');
```

donne une p-value de $0,64$, on accepte largement le test à 5%.

8) On a la relation suivante (densité conditionnelle)

$$p^\theta(z_1,\dots,z_n) = \frac{\mathcal{L}_c^\theta(x_1,z_1,\dots,x_n,z_n)}{\mathcal{L}^\theta(x_1,\dots,x_n)}$$

Donc, en prenant les log et les espérances pour le modèle θ', il vient,

$$\log \mathcal{L}^\theta(x_1,\dots,x_n) = Q(\theta,\theta') - H(\theta,\theta')$$

Ce qui nous donne une différence entre deux log-vraisemblance successives (après une itération de l'algorithme), en prenant $(\theta,\theta') = (\theta_{k+1},\theta_k)$ et $(\theta,\theta') = (\theta_k,\theta_k)$ dans la relation précédente, et en effectuant la soustraction,

$$\log \mathcal{L}^{\theta_{k+1}}(x_1,\dots,x_n) - \log \mathcal{L}^{\theta_k}(x_1,\dots,x_n) = \big(Q(\theta_{k+1},\theta_k) - Q(\theta_k,\theta_k)\big) - \big(H(\theta_{k+1},\theta_k) - H(\theta_k,\theta_k)\big)$$

Or on sait (étape M) que $\big(Q(\theta_{k+1},\theta_k) - Q(\theta_k,\theta_k)\big) \geqslant 0$. Calculons

$$\big(H(\theta_{k+1},\theta_k) - H(\theta_k,\theta_k)\big) = \mathbb{E}_{\theta_k}[\log \frac{p^{\theta_{k+1}}}{p^{\theta_k}} | X_1 = x_1,\dots,X_n = x_n]$$

On utilise alors l'inégalité de Jensen pour conclure que

$$\mathbb{E}_{\theta_k}[\log \frac{p^{\theta_{k+1}}}{p^{\theta_k}} | X_1 = x_1,\dots,X_n = x_n] \leqslant \log \mathbb{E}_{\theta_k}[\frac{p^{\theta_{k+1}}}{p^{\theta_k}} | X_1 = x_1,\dots,X_n = x_n] = 0$$

On trouvera cette preuve ainsi que beaucoup d'autres compléments et exemples concrets dans [2].

Références

[1] P. Doukhan. Estimation et introduction aux tests. Décembre 2004.

[2] G.J. McLachlan and T. Krishnan. *The EM algorithm and extensions.* Wiley, New York, 1997.

[3] R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26 :195–239, 1984.