Implémentation sur DSP d'un démodulateur numérique du signal aéronautique VOR

Projet d'option de 2ème année de cycle d'Ingénieur à l'ENAC

THOMAS Johan
LUCAS Fabrice

Responsable de projet :
M. Jean-Marc Louis

Sommaire

I. INTRODUCTION

Les récepteurs au sol de contrôle du signal VOR[1] réalisent actuellement un traitement analogique du signal basse fréquence (BF) VOR. Un traitement numérique serait préférable car il permettrait d'obtenir une meilleure stabilité ainsi qu'une meilleure reproductibilité des mesures et surtout d'inclure ces récepteurs dans une chaîne d'acquisition automatique de mesures pilotée par PC.

Ce projet montre qu'il est possible de réaliser un tel traitement tout en conservant l'ordre de précision des récepteurs analogiques. L'outil qui nous a permis d'obtenir un tel résultat est le processeur de signaux (DSP) 21020 de la famille ANALOG DEVICE : la quantification (sur 8 bits) qu'il effectue lors de l'échantillonnage et la précision des calculs qu'il opère (format des données : 32 bits), éléments clés lorsque l'on recherche la précision, ont permis de satisfaire au cahier des charges. D'autres caractéristiques (choix possibles pour la fréquence d'horloge…) que nous expliciterons dans ce rapport nous ont conduit à le choisir. Toutefois, nous montrerons que des qualités supérieures, notamment en terme de quantification, permettraient d'aboutir à des résultats supérieurs.

[1] VHF Omnirange : radiophare VHF omnidirectionnel

II. PRESENTATION DU RECEPTEUR

1. Rôle et principe de fonctionnement du VOR

Le VOR est le système de mesure d'azimut normalisé par l'OACI[2] pour la navigation aérienne. Cette aide radioélectrique au sol rayonne dans tout azimut un signal indiquant à l'utilisateur son relèvement magnétique pris à la station d'émission référencée au Nord Magnétique. Plus précisément, le VOR émet une porteuse VHF modulée de façon à transmettre simultanément et indépendamment deux signaux BF 30 Hz :

- un signal de référence (30 REF) dont la phase est identique quel que soit l'azimut d'émission.
- un signal variable (30 VAR) dont le déphasage $\Delta\varphi$ par rapport au précédent est égal à l'azimut α de la direction d'émission

Or une bonne séparation des deux signaux dans le récepteur à bord nécessite deux modulations différentes. C'est pourquoi l'OACI a retenu, dans le cas du VOR Conventionnel (VOR C), la méthode suivante :

- le 30 VAR est transmis directement en modulation d'amplitude de la porteuse à ω
- le 30 REF est transmis au moyen d'une sous porteuse à $f_0 = 2\pi\omega_0 = 9960$ Hz modulée en fréquence par le 30 REF. Cette sous porteuse modulée est alors transmise à son tour par modulation d'amplitude de la porteuse à ω

Le VOR transmet également, par modulation d'amplitude de la porteuse à ω, des signaux d'identification de la station et de radiophonie.

2. Signal reçu en entrée du récepteur du signal VOR

Le signal reçu en entrée d'un récepteur VOR peut se mettre sous la forme suivante :

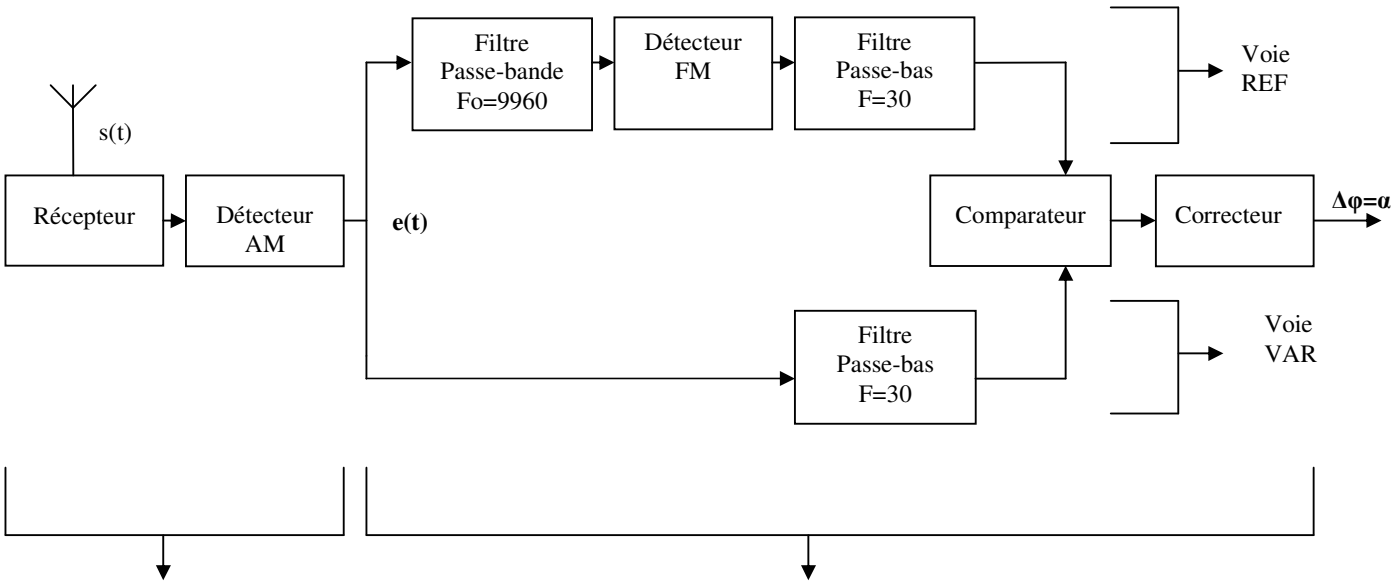
$$\mathbf{s(t) = E_p\,[1+m_1 * \cos(\omega_0 t + n\sin\Omega t) + m_2 * identification + m_3 * phonie + m_4 * \cos(\Omega t - \alpha)\,]\,\cos(\omega t)}$$

[2] Organisation de l'Aviation Civile Internationale

Les valeurs numériques des différents paramètres normalisés par l'OACI sont :

- E_p : amplitude du champ électrique, en limite de couverture E_p = 90 µV/m
- ω : pulsation de la porteuse, dans la bande [108-118 Mhz]
- m_1 : taux de modulation de la sous porteuse, 30% nominal
- ω_0 : pulsation de la sous porteuse, 9960 Hz (±1%)
- Ω : pulsation des signaux 30 REF et 30 VAR, 30 Hz (±1%)
- n : indice de modulation, 16 ± 1, soit une excursion en fréquence Δf_0 de l'ordre 480 Hz (16*30)
- m_2 : taux de modulation de l'identification, 10 ou 5 % (tonalité 1020 Hz découpée suivant le code MORSE)
- m_3 : taux de modulation crête de la phonie,<30% (bande spectrale limitée à 300-3000 Hz)
- m_4 : taux de modulation du 30 VAR, pour le VOR C, m_4 = 30% pour le VOR C

3. Synoptique d'un récepteur VOR



La partie HF étant déjà réalisée, notre projet a consisté à traiter l'entrée **e(t)** de la partie BF afin d'extraire **Δφ** déphasage entre le 30 REF et le 30 VAR avec une précision de 0.1°.

4. Etude de l'entrée de la partie BF

Le signal après détection peut se mettre sous la forme suivante :

$$\mathbf{e(t) = E_p\,[1+m_1{*}\cos(\omega_0 t+n\sin\Omega t) + m_2{*}identification + m_3{*}phonie + m_4{*}\cos(\Omega t - \alpha)\,]}$$

On peut également donner une représentation fréquentielle de ce signal. Le spectre du signal d'identification se compose d'une raie à 1020 Hz et le spectre de la phonie s'étend sur la bande [300 3000Hz]. Le signal 30VAR ($E_p\ m_4{*}\cos(\Omega t - \alpha)$), signal sinusoïdal pur à la fréquence 30 Hz, possède un spectre formé d'une raie à 30 Hz. La représentation spectrale du dernier terme $e(t)_{S.P} = E_p \cos(\omega_0 t+n\sin\Omega t)$ est a priori moins immédiate.

Remarquons tout d'abord que ce signal est périodique de fréquence 30 Hz : en effet, Ω =30 Hz et ω_0 = 9960 = 30*332 Hz est un multiple de Ω. Le spectre est donc un spectre de raies espacées de 30 Hz. On peut montrer qu'elle sont en nombre infini. Sous Matlab, nous avons pu visualiser ce spectre (Figure 1).

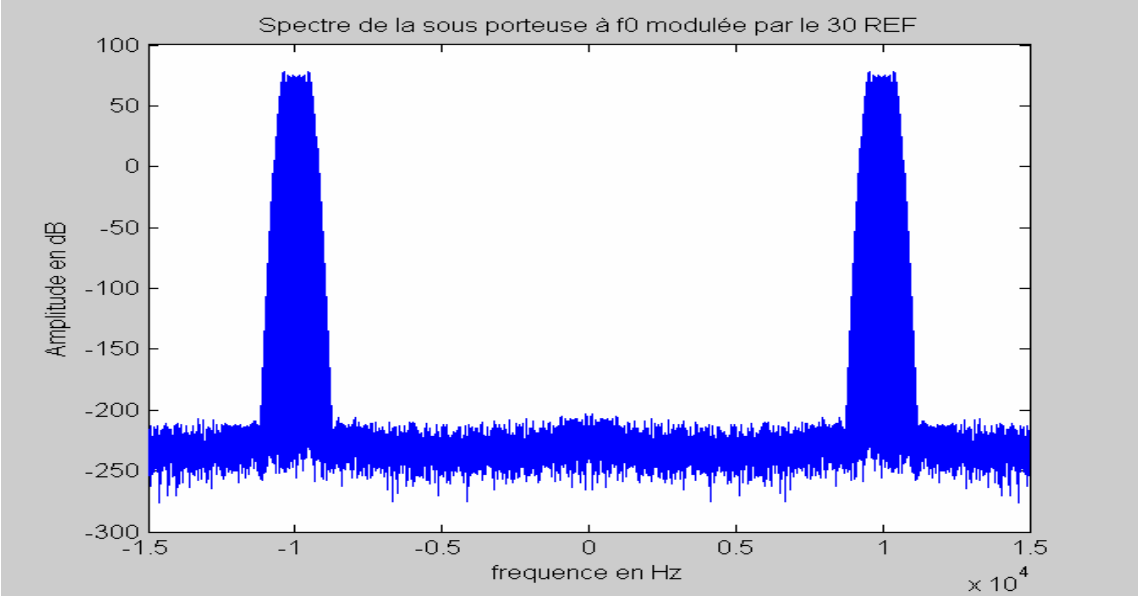


Figure 1

En zoomant sur la partie droite du spectre ("fréquences positives"), c'est-à-dire sur le spectre du signal analytique au coefficient multiplicatif ½ près, on peut observer les raies (Figure2) avec un espacement de 30 Hz (Figure3).

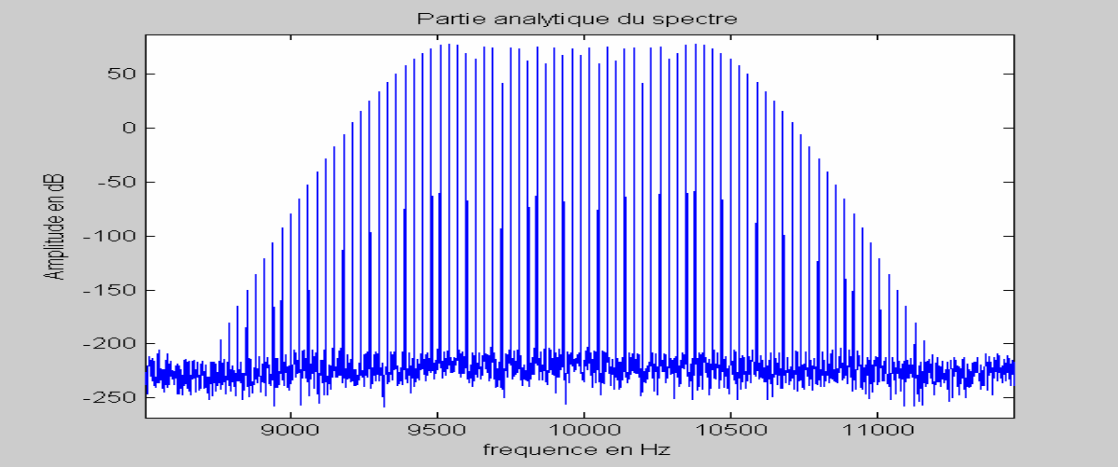


Figure 2

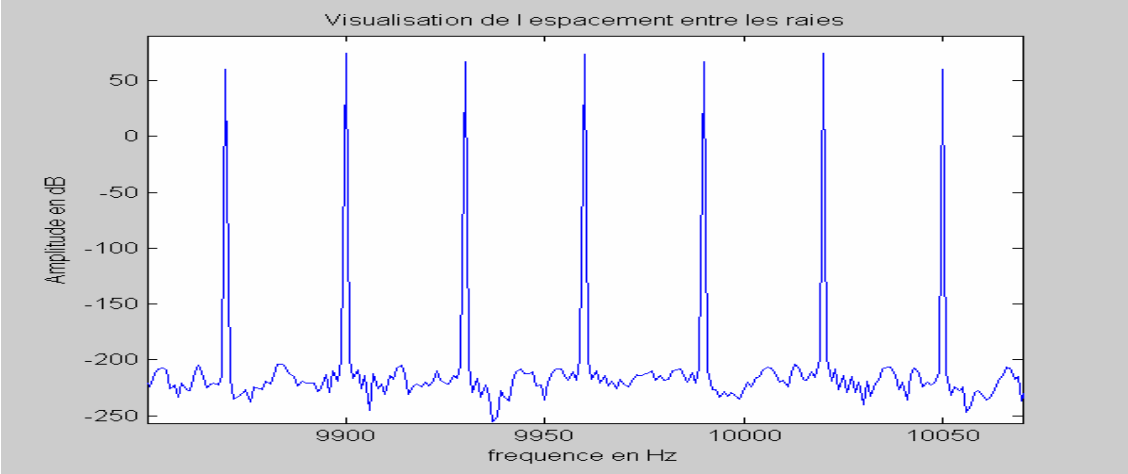
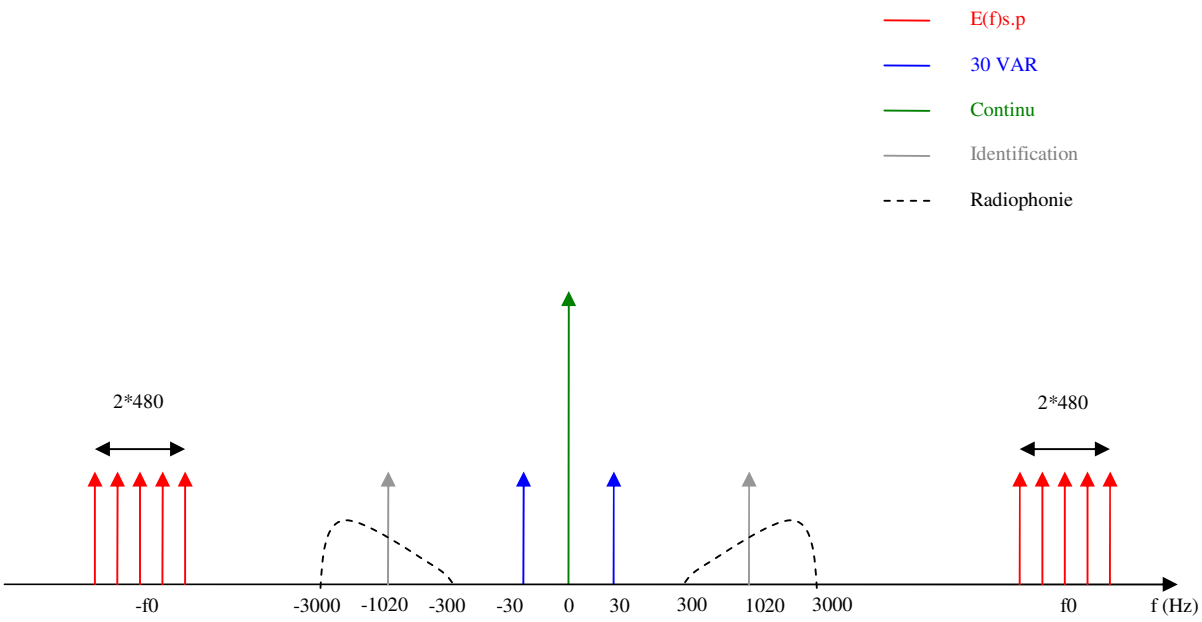


Figure 3

On constate que le signal analytique est symétrique autour de la raie à 9960 Hz (30*332).Il s'évanouie rapidement pour f > 10440 Hz (10440=9960+16*30=9960+480) à droite et pour f< 9480 Hz (9480=9960-16*30 = 9960-480). Nous considérerons donc que le signal analytique a un spectre borné compris entre 9480 et 10440 Hz.

Remarque : On peut aboutir à cette hypothèse en raisonnant avec la notion de fréquence instantanée. A l'instant t, $e(t)_{S.P}$ possède une fréquence instantanée f(t) avec $f(t) = 1/(2\pi) *(\omega_0+n\Omega\cos\Omega t) = f_0 +\Delta f *\cos\Omega t$ où $\Delta f = n*30 \approx 16*30= 480$ Hz. On voit donc que le signal est plus ou moins contracté à l'instant t et sa fréquence instantanée oscille entre et $f_0 - \Delta f$ et $f_0 +\Delta f$ i.e entre 9960-480 Hz et 9960+480 Hz dans le cas idéal (cf paramètres normalisés par l'OACI) On convient donc en pratique que le spectre se limite à la bande [9960-480 9960+480 Hz].

Spectre du signal reçu et à traiter en entrée de la partie BF



Données propres au VOR de l'ENAC :

- Fréquence de la station : 110.4 MHz
- Puissance de l'émetteur : 50 W
- Capteur de contrôle placé à 30 m de la station
- Niveau reçu par le récepteur : -26 dBm
- Niveau après détection :

  * 2V pour la composante continue
  * 0.6 V crête pour la composante à 9960 Hz
  * 0.6 V crête pour la composante à 30 Hz
  * 0.2 V pour la composante à 1020 Hz

Nous avons pris ce VOR pour référence dans notre étude. De plus, pour réaliser les tests, nous n'avons pas généré les signaux d'identification et de phonie car seuls le déphasage $\Delta\varphi$ entre le 30 REF et le 30 VAR nous intéresse. Désormais, nous nous intéresserons donc à un signal du type :

$$\mathbf{e(t) = [2+0.6*\cos(\omega_0 t+n\sin\Omega t) +0.6\cos(\Omega t+ \Phi)] \quad avec\ \Phi\ entre\ 0\ et\ 360^\circ\ (\Phi=-\alpha)}$$

Nous avons simulé son allure sous Matlab pour $\Phi = 20\ ^\circ$ (Figure 4).

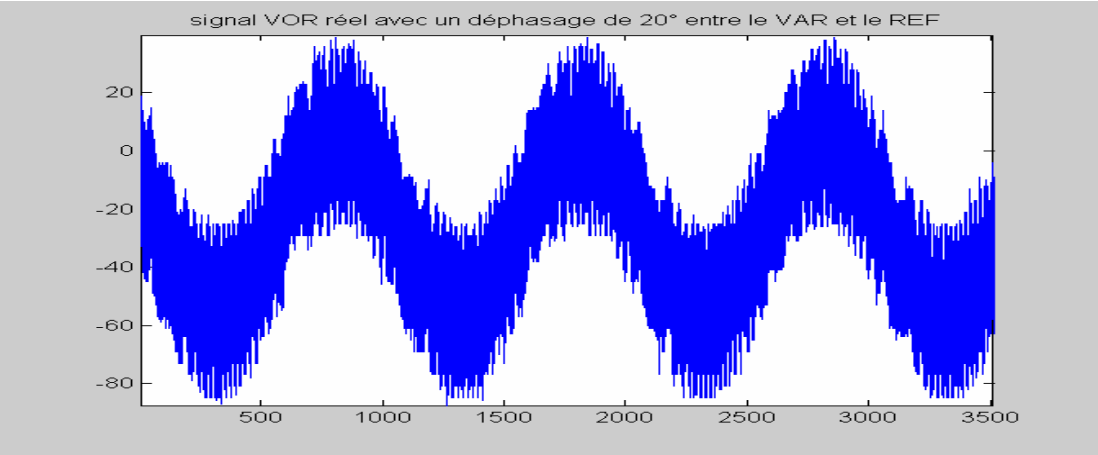
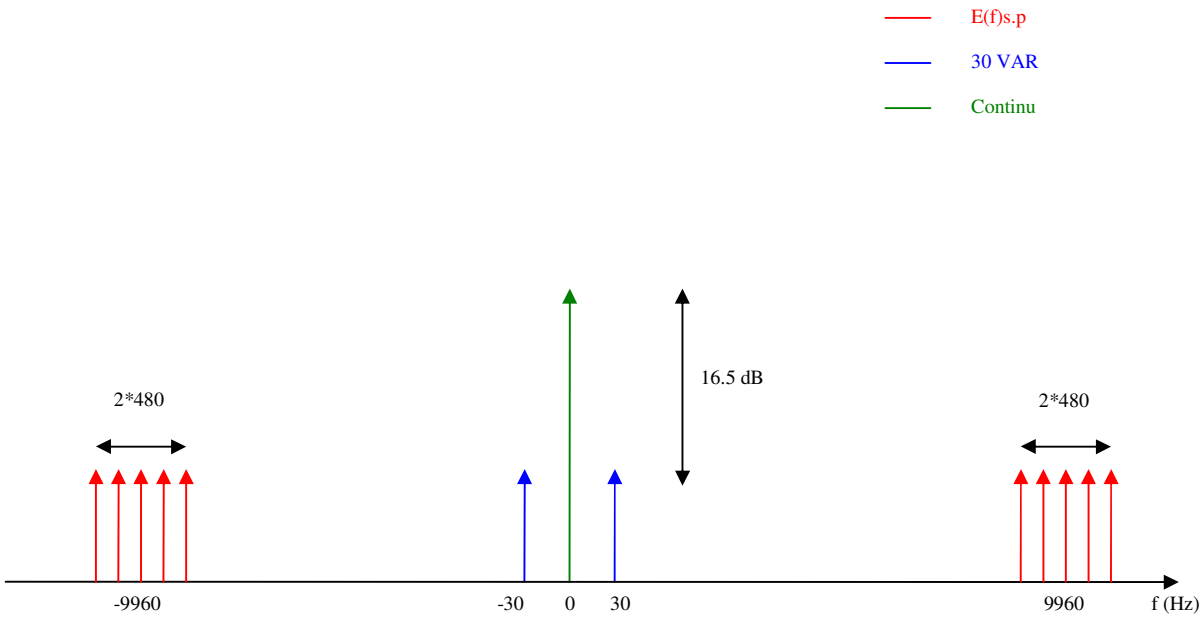


Figure 4

Représentation schématique du spectre E(f)de e(t)



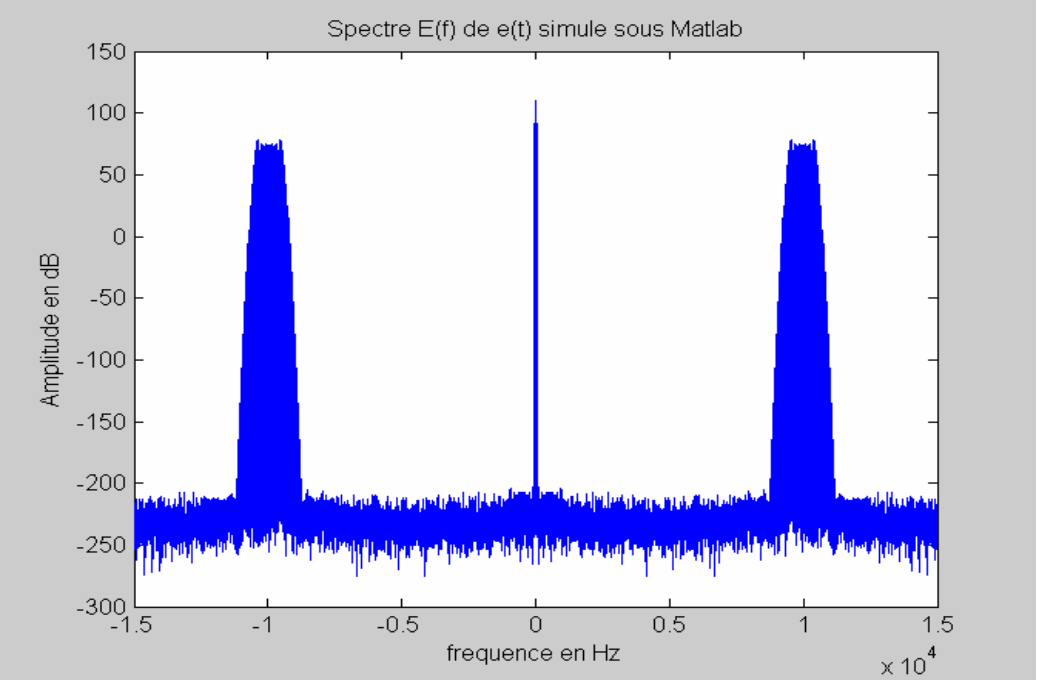


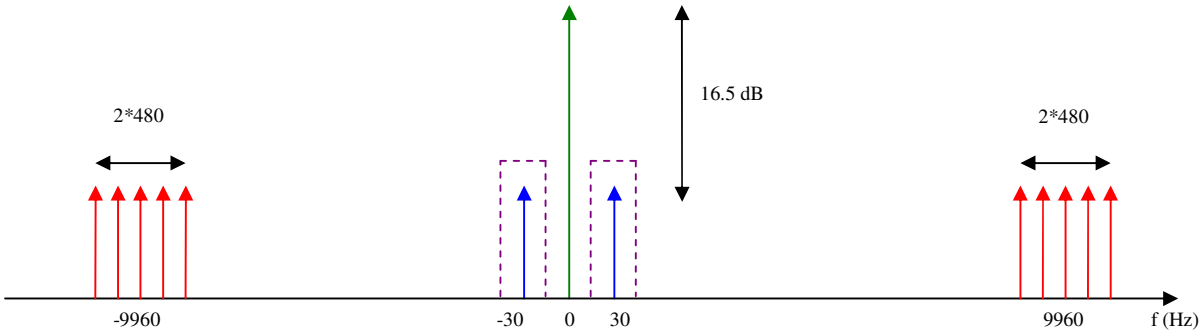
Figure 5

5. Choix de la fréquence d'échantillonnage

D'après le théorème de Shannon, on sait qu'il faut choisir une fréquence d'échantillonnage fe supérieure à 2 fois la fréquence maximale du spectre E(f) pour que l'opération d'échantillonnage soit réversible. La fréquence maximale est égale à fréquence maximale de la sous porteuse modulée par le 30 REF à savoir $f_0 + \Delta f = f_0 + n*\Omega/(2\pi)$. Or dans le pire des cas, f_0,n et $\Omega/(2\pi)$ valent 10059 Hz (9960*(1.01)), 17 (16+1), 30.3 (30*1.01). Donc au pire la fréquence maximale est de 10574. Nous avons choisi une fréquence d'échantillonnage **fe = 30 kHz**.

III. DEVELOPPEMENT DES ALGORITHMES

1. Extraction du signal variable (30 VAR)



Pour obtenir le signal VAR, il suffit de filtrer la raie à 30 Hz. Toutefois, il est difficile de générer sous Matlab un filtre passe bande à 30 Hz sans déborder sur le continu, à moins d'avoir un ordre très élevé ce qui entraînerait de longs temps de calculs. On adopte alors la méthode suivante. On utilise un filtre RIF[3] passe bas d'ordre 100 à phase linéaire et de fréquence de coupure 90 Hz (Figure 6)). Il sera réalisé par la fonction fir1 de Matlab.

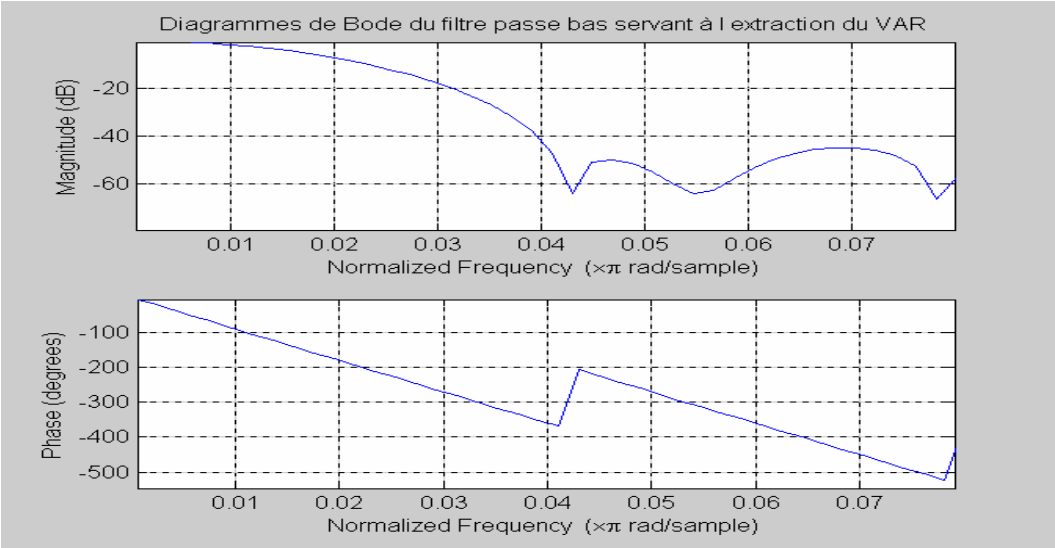


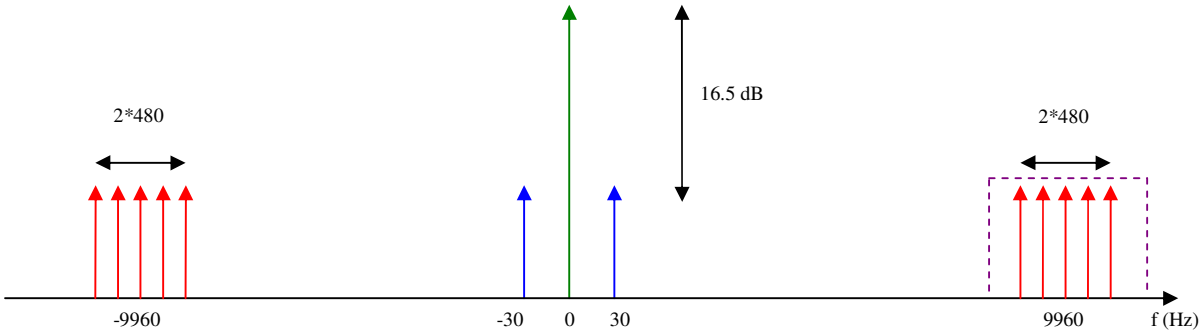
Figure 6

[3] Filtre à Réponse Impulsionnelle Finie

A la sortie de la voie VAR, on obtient la composante continue et le 30 VAR déphasé de Φ_{PB} , déphasage appliqué à une sinusoïde à 30 Hz . On calcule ensuite la moyenne moy_var de ce signal : lors du calcul du déphasage, on travaillera sur les valeurs obtenues auxquels on retranchera la moyenne.

2. Extraction du signal de référence (30 REF) modulé

Pour obtenir le signal REF il faut filtrer la bande spectrale [9960-480,9960+480]. Le signal initial est de la forme $e(t) = 2+0.6*\cos(\omega_0.t+n\sin(\Omega t))+0.6*\cos(\Omega.t+\Phi)$.Or le signal portant l'information est en fait l'argument du signal modulé en fréquence. La méthode employée consiste à filtrer les composantes fréquentielles positives autour de 9960 Hz puis à récupérer l'argument grâce à la fonction arctan appliquée au couple (partie imaginaire , partie réelle).



Nous avons choisi un filtre de Hilbert (Figure 7) autour de +9960 Hz. Celui-ci est obtenu par translation d'un RIF passe bas. Pour effectuer la translation de ce filtre, il suffit de récupérer les coefficients du passe-bas et de les multiplier par $\exp(j.2\pi kf_0/fe.)$ où k est l'indice du coefficient. En effet :

$$H_{hilbert}(f)=H_{pb}(f-9960)=\sum ak.\exp(-j.2\pi k(f-9960)/fe.)$$

$$=\sum ak'.\exp(-j2\pi.kf/fe.)$$

où $ak'=ak.\exp(j2\pi.k9960/fe.)$

Le filtre (ak) choisi est un filtre RIF passe bas d'ordre 100 à phase linéaire et de fréquence de coupure 480 Hz (Figure 8). Il sera réalisé par la fonction fir1 de Matlab.

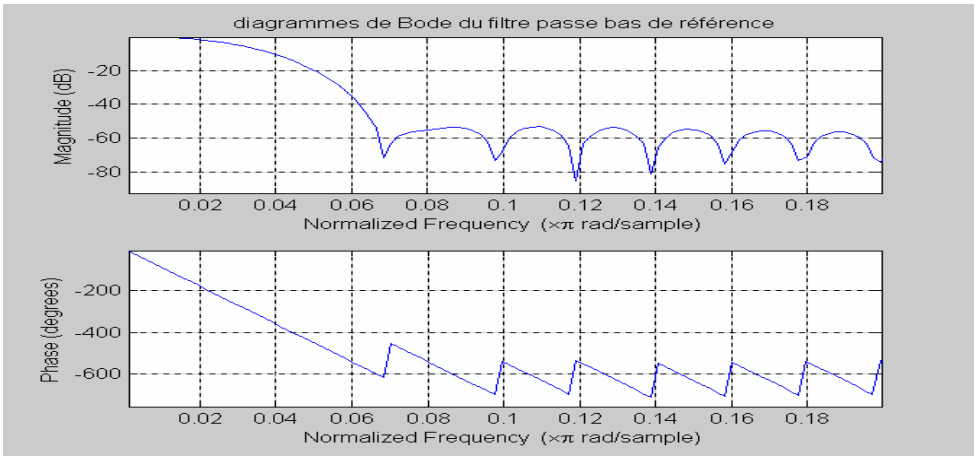


Figure 7

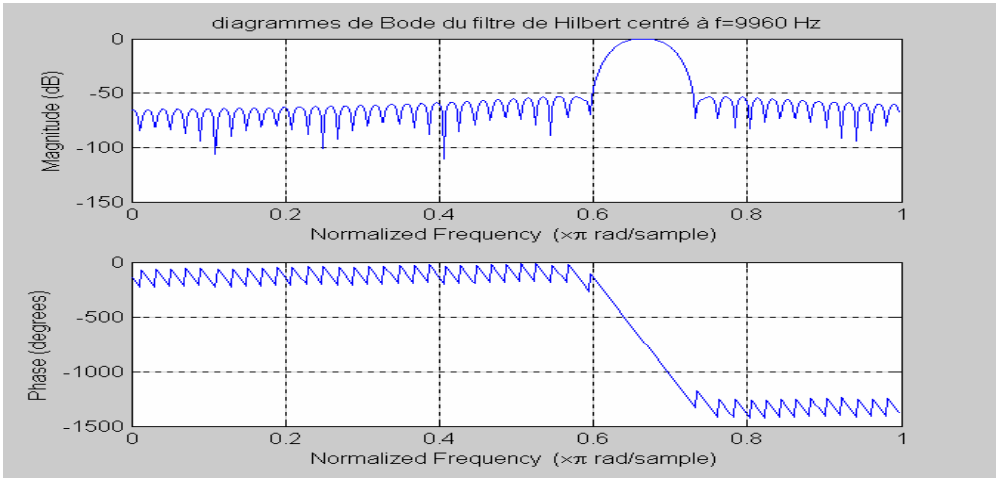


Figure 8

Le résultat de ce filtrage est de la forme $0.6*\exp[\, j\, (\omega_0.t+n.\sin(\Omega.t+\phi1)+\phi2)\,]$ où $\phi1$ et $\phi2$ sont des phases caractéristiques du filtre de Hilbert qui seront explicitées par la suite.

Il reste ensuite à récupérer l'argument de l'exponentielle complexe : $\omega_0.t+n.\sin(\Omega t+\phi1)+\phi2$

En pratique la fonction arctan2 de Matlab est à valeurs dans $[-\pi,\pi]$ et la fonction temporelle obtenue présente donc des discontinuités de $+\pi$ à $-\pi$ et de $-\pi$ à π. (Figure 9)

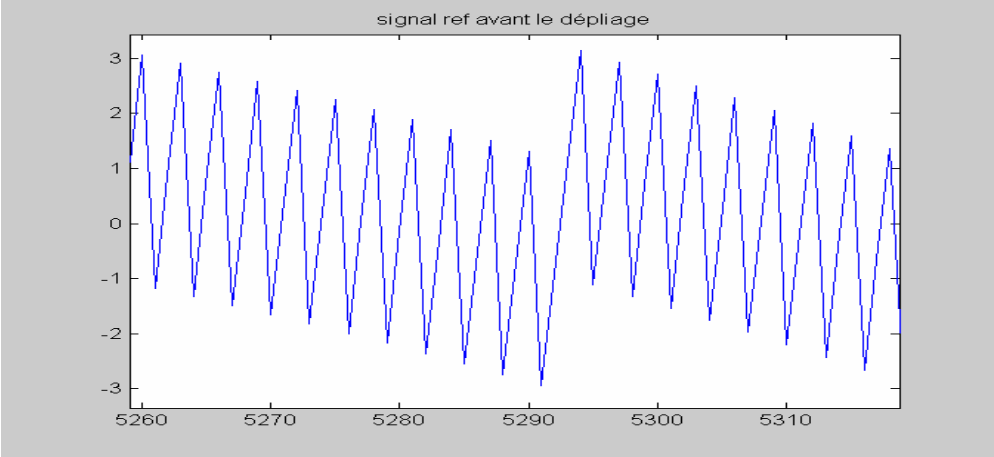


Figure 9

La solution retenue est de détecter ces discontinuités en comparant 2 valeurs consécutives et :
-d'ajouter 2π si $s(n+1)-s(n) < \pi$
-de soustraire 2π si $s(n+1)-s(n) > \pi$

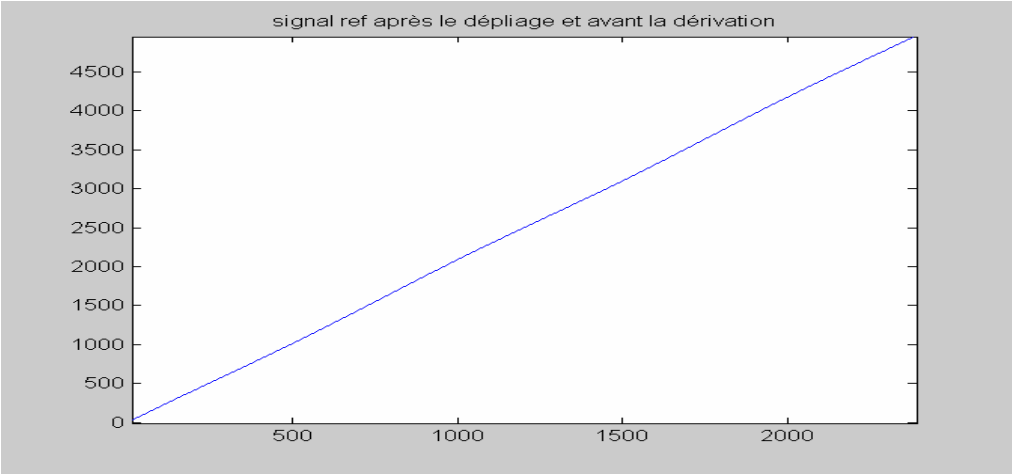


Figure 10

Enfin en dérivant l'expression obtenue on aboutit à :
$s(t) = \omega 0 + n.\Omega.\cos(\Omega.t+\phi 1)$, signal sinusoïdal à 30 Hz de moyenne $\omega 0$.

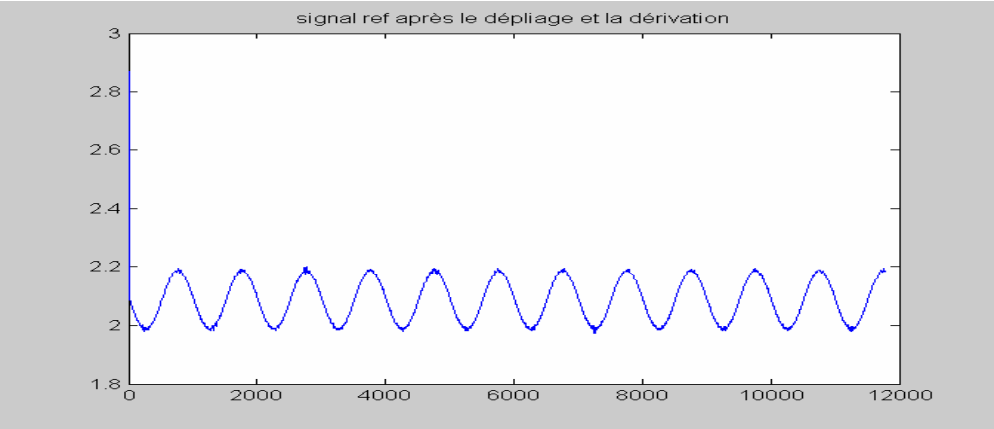


Figure 11

En pratique, les opérations de dépliage et de dérivation seront réalisées en même temps grâce à l'algorithme :

```
for k=2..longueur(s)

    if s(k)-s(k-1)<-pi
       s(k-1)=s(k)-s(k-1)+2*pi;
     else
          if s(k)-s(k-1)>pi
                s(k-1)=s(k)-s(k-1)-2*pi;
          else
              s(k-1)=s(k)-s(k-1);
          end
    end
 end
```

On calcule la valeur moy_ref moyenne de s(t), ce qui permet de récupérer la valeur de f_0. On note f9960 la valeur mesurée de f_0 qui n'interviendra pas dans la correction de phase. Ensuite l'application du même filtre passe-bas que celui de la voie VAR permet de lisser la courbe (Figure 12) et apporte un déphasage égal à Φ_{PB} . Lors du calcul du déphasage, on travaillera sur les valeurs obtenues en auxquels on retranchera la moyenne.

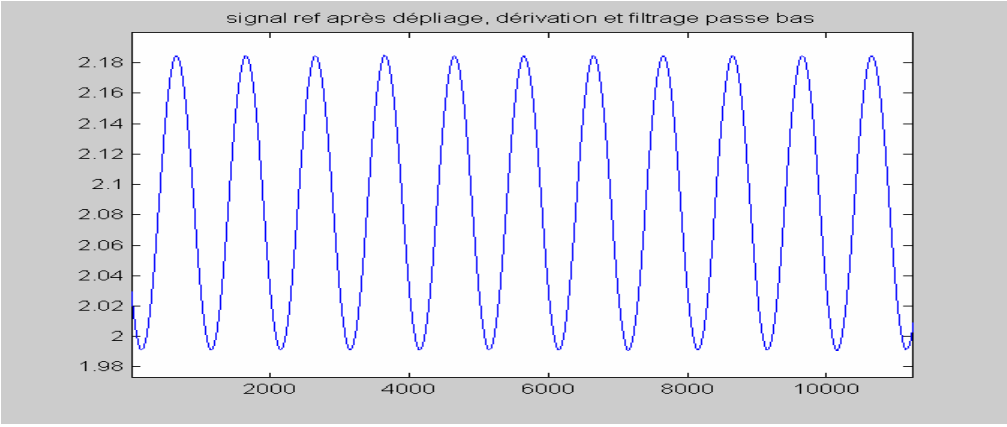


Figure 12

3. Calcul du déphasage

Pour mesurer le déphasage entre les deux signaux reçus, la solution consiste à détecter les zéros de chaque signal(filtré et centré) sur un temps donné et à comparer leurs positions respectives.

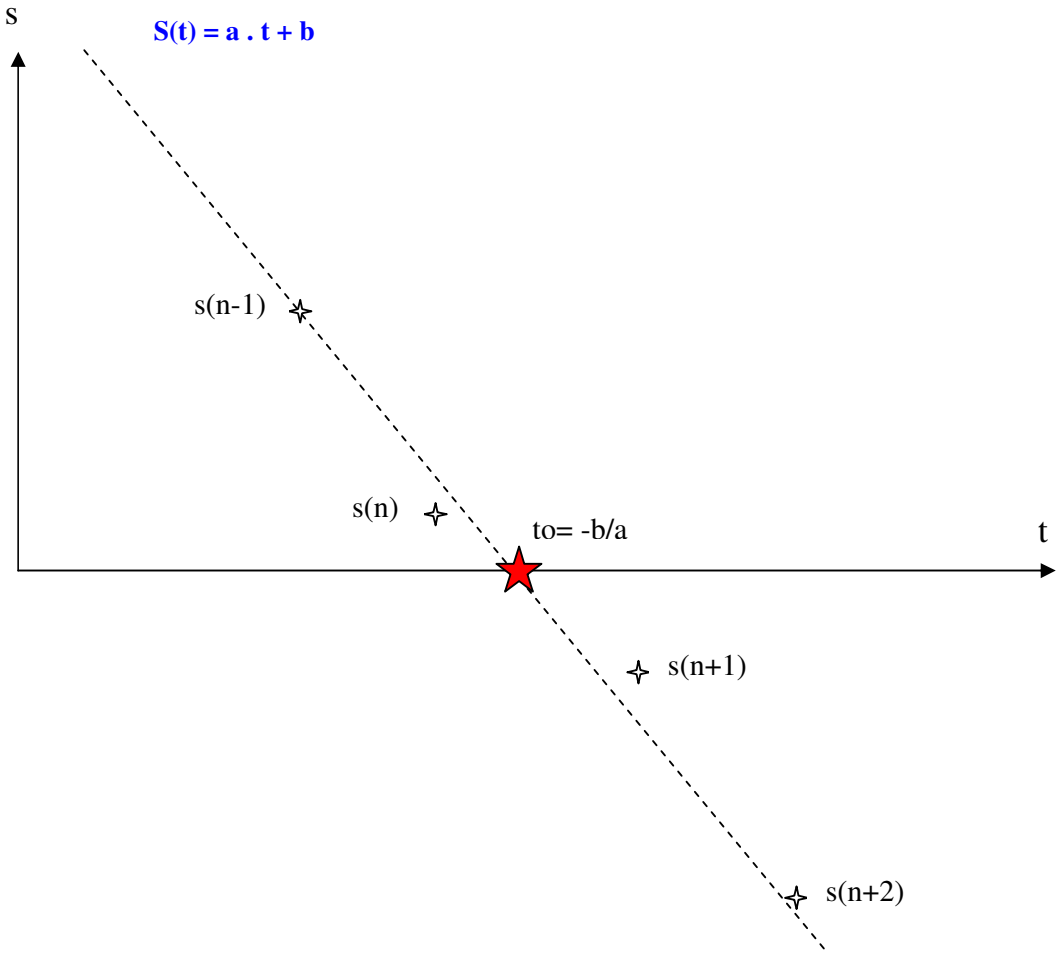
La détection aboutit à deux séries de N valeurs temporelles correspondant aux positions des zéros du REF et du VAR : tref[1 :N] et tvar[1 :N]

Pour cela on détecte les changements de signe du signal puis pour minimiser l'effet du bruit on effectue une régression linéaire sur les 4 points entourant le changement de signe.

Pour un ensemble de points (xi,yi) la régression linéaire donne une droite affine y = a.x+b avec :

$$a = \frac{N\sum x_i y_i - \sum x_i \sum y_i}{N\sum x_i^2 - (\sum x_i)^2}$$

$$b = \frac{\sum y_i \sum x_i^2 - \sum x_i \sum x_i y_i}{N\sum x_i^2 - (\sum x_i)^2}$$



Voici l'algorithme utilisé :

```
i=0, n=0;
while n<N;

   if  Sref(i+1)*(Sref(i)<0;

      A=(i-1)*Sref(i-1)+i*Sref(i)+(i+1)*Sref(i+1)+(i+2)*Sref(i+2);
      B=Sref(i-1)+Sref(i)+Sref(i+1)+Sref(i+2);
      tref(n+1)=( ( ((4*i+2)*A-B*(4*i^2+4*i+6)) /(4*A-B*(4*i+2)) ) /fe);
      n=n+1;
      i=i+round(fe/fb/2)-50
    end

   if Sref(i)==0;
      tref(n+1)=(i/fe);
      n=n+1;
      i=i+round(fe/fb/2)-50
   end

   i=i+1;
end
```

i correspond au pointeur de position sur l'axe temporel
n comptabilise le nombre de zéros
A et *B* sont deux calculs servant à la régression
`i=i+round(fe/fb/2)-50` permet de positionner le pointeur peu avant le zéro suivant

Ensuite pour le calcul du déphasage, on doit tenir compte des fronts montants ou descendants. Si les deux signaux sont positifs (ou négatifs) à l'instant initial, le premier zéro du REF et du VAR sont deux zéros descendants (ou montants) et le déphasage sera calculé en faisant la moyenne des différences tref(i)-tvar(i) . Si les deux signaux ne sont pas de même signe à l'instant initial, on fera la moyenne des différences tref(i+1)-tvar(i)

En pratique :

```
if Sref(1)*Svar(1)>0
   for i=1:N;
      {
      dt(i)= tref(i)-tvar(i);
      }
   dT=mean(dt);
else
   for i=1:N-2;
      {
      dt(r)= tref(i+1)-tvar(i);
      }
   dT=mean(dt);
end
```

Les premiers résultats ont mis en évidence l'avantage de choisir N pair. En effet les demi-périodes ne sont pas strictement égales car les signaux traités ne sont pas exactement centrés.

Pour convertir la différence temporelle en déphasage, on multiplie le résultat par $2\pi.f30$, où f30 est la fréquence mesurée des signaux 30 REF et 30 VAR. On obtient alors l'avance de phase brute(non corrigée) du 30 VAR par rapport au 30 REF.

Remarque : la valeur f30 s'extrait des séries de zéros . En effet, en faisant la moyenne des différences tref(i+1)-tref(i) , on obtient une estimation de la demi-période du 30 REF :

```
for i=1:N-2
   dt(i)=tref(i+1)-tref(i);
end
Fref = 1 /(mean(dt)*2)
```

En pratique, seule la fréquence du 30 REF sera mesurée car la modulation de fréquence offre une meilleure protection contre le bruit. Cette fréquence f30 est employée pour la correction de phase du filtre de Hilbert .

Remarque : dans le programme C final, nous avons fait le choix de travailler avec les valeurs Sref-moy_ref et Svar –moy_var dans l'algorithme de recherche des zéros plutôt que de centrer préalablement les signaux car on évite ainsi l'exécution de 2 boucles de 11500 itérations (gain de temps lors de l'exécution du programme en C) .

4. Correction du déphasage

- Déphasage appliqué au 30 VAR : Φ_{PB} dû au filtre passe bas de la voie VAR
- Déphasage appliqué au 30 REF : après application de l'algorithme de dépliage- dérivation, on obtient le signal $s(t+te/2)=2\pi.f_0 +n.\Omega/(2\pi).\cos(\Omega.(t+te/2)+\phi1)$ car la dérivation translate le signal de te/2 vers la gauche. En effet, à chaque itération k on calcule la dérivée en k-1/2 (s(k)-s(k-1)) et on la stocke s(k-1). Enfin, ce signal est filtré par le même filtre passe-bas que celui de la voie VAR qui lui apporte un déphasage égale à Φ_{PB}. Au total, en sortie de la voie REF, on obtient le 30 REF déphasé de $\phi1+\Omega.te/2+ \Phi_{PB}$.

Au déphasage calculé $\phi var - \phi ref$, il faut donc ajouter le correctif **$\Delta\Phi = \phi1+\Omega.te/2$**.

Reste à déterminer $\phi1$. On étudie pour cela, le filtrage de Hilbert. On note $H_{hilbert}(f)$et $H_{pb}(f)$ les gabarits respectifs du filtre de Hilbert et du passe bas RIF d'ordre 100 (101 coefficients),à phase linéaire et de fréquence de coupure 480 Hz, dont il est issu. On peut écrire $H_{pb}(f)$ sous la forme $H_{pb}(f) = A(f)*\exp(-j2\pi f/fe*t_0)$ où vaut $t_0 = 100/(2fe)$, $A(f)\cong 1$ dans la bande passante [-480 Hz 480 Hz] et $A(f)\cong 0$ ailleurs. Or $H_{hilbert}(f)=H_{pb}(f- 9960)$ donc :

$H_{hilbert}(f) = A(f- 9960)*\exp(-j2\pi(f- 9960)*t_0)$
$H_{hilbert}(f) = A(f- 9960)*\exp(-j2\pi f*t_0)* \exp(j2\pi* 9960*t_0)$

En entrée, on a e(t) de transformée de Fourier E(f). On note x(t) le signal de sortie et X(f) sa transformée de Fourier.

$X(f) = H_{hilbert}(f)*E(f)$.

$H_{hilbert}(f)$ étant nul en dehors de la bande B = [9960-480 Hz 9960+480 Hz] et si on note $E_B(f)$ la fonction qui vaut E(f) dans B et 0 partout ailleurs, on obtient :

$X(f) = H_{hilbert}(f) * E_B(f)$.
$X(f) = A(f- 9960)*\exp(-j2\pi f*t_0)* \exp(j2\pi* 9960*t_0)* E_B(f)$
$X(f) = \exp(j2\pi* 9960*t_0)\exp(-j2\pi f*t_0)* E_B(f)$ (1)

Or $E_B(f)$, spectre E(f) restreint à la bande B, est en fait la transformée de Fourier du signal analytique du 30 REF modulé (au coefficient multiplicatif ½ près), c'est à dire de $e_B(t) = 0.6*\exp[j (\omega_0.t+n.\sin(\Omega.t))$.

Si on exprime la relation (1) en temporel, on obtient :

$x(t) = \exp(j2\pi* 9960*t_0)* e_B(t- t_0)$
$x(t) = 0.6*\exp[j (\omega_0.(t-t_0)+n.\sin(\Omega.(t-t_0)))+ j2\pi* 9960*t_0]$.

Donc : $\phi 1 = - \Omega.t_0$
$\phi 1 = - \Omega*100/(2*fe)$

Conclusion : le correctif $\Delta\Phi$ à appliquer vaut $-\Omega*100/(2fe)+\Omega.te/2$ radians. En pratique, on applique **un correctif $\Delta\Phi$ en degré de -180*f30*100/fe + f30*2*180/(2fe).**

5. Résultats obtenus

Voici les courbes d'erreurs obtenues sous Matlab à partir de signaux simulés plus ou moins quantifiés. Celles-ci prédisent une erreur finale acceptable. On remarque qu'en augmentant la quantification l'erreur diminue de façon significative.(Figure 13 et 14)

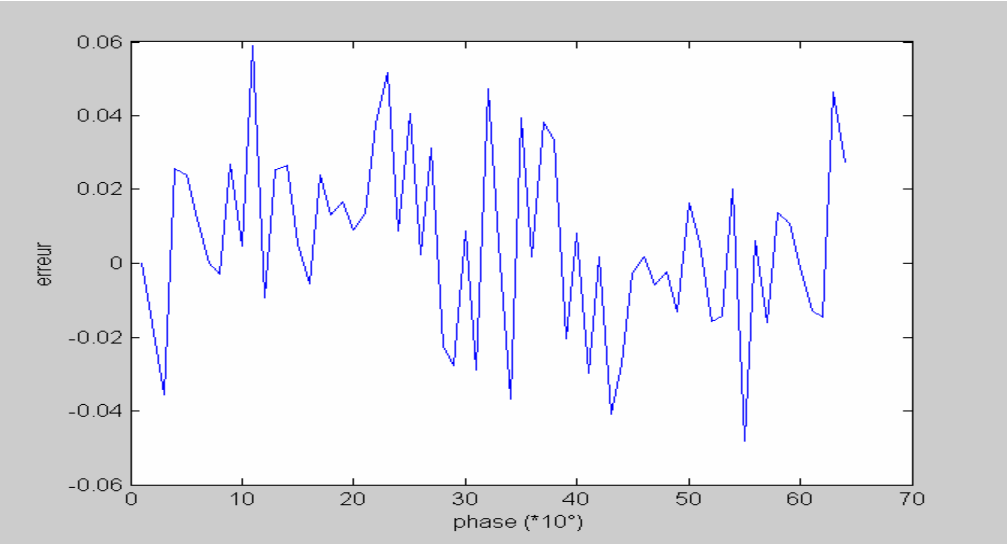


Figure 13: prévision des erreurs obtenues avec notre DSP (quantification sur 8 bits)

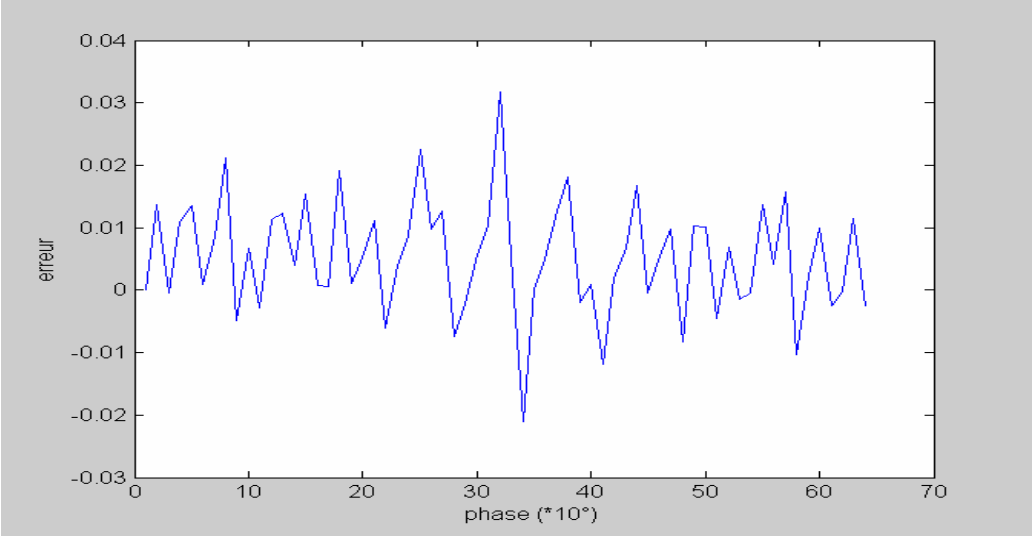


Figure 14: prévisions des erreurs obtenues avec une quantification sur 16 bits

IV. TRANSPOSITION DU PROGRAMME SUR DSP

1. Choix du processeur de signaux

Le signal e(t) varie entre 0.8 V et 3.2 V autour de la valeur moyenne 2V. Parmi les DSP dont nous disposions à l'ENAC, seul l'ANALOG DEVICE 21020 peut accepter des signaux évoluant dans cette plage de tensions et contenant une composante continue. De plus, il était le seul à pouvoir monter jusqu'à 30 kHz pour la fréquence d'échantillonnage.

2. Structure du programme

Le programme est constitué d'une partie principale , d'une procédure d'interruption et de trois fonctions (moyenne, filtrage passe bas, recherche de zéros)

Le programme principal configure le timer, autorise ou non l'interruption d'échantillonnage, effectue le traitement du signal VAR et mesure le déphasage. Il doit aussi gérer un buffer tournant qui permet de moyenner les mesures de déphasage.

La procédure d'échantillonnage capte la valeur en entrée, duplique le signal pour pouvoir effectuer deux traitements distincts sur le REF et le VAR et traite le signal REF pour diminuer la durée des traitements ultérieurs dans le main. La durée de traitement doit en effet être minimisée afin de permettre l'utilisation du programme en temps réel.

3. Le programme principal (MAIN)

Après les déclarations de variables le programme principal configure la procédure d'interruption et le timer. Celui-ci est réglé à 30 Khz au moyen de l'argument 1100 de la fonction timer_set. En effet 1100*30Khz = 33Mhz qui est la fréquence d'horloge du DSP.

Ensuite une boucle infinie est ouverte qui permettra aux mesures de phases de se succéder.

Au début de cette boucle, les interruptions sont autorisées pendant 11500 acquisitions qui correspond au nombre d'échantillons souhaité.

Pendant ces interruptions le filtrage et le traitement du REF sont effectués et la première opération qui suit consiste à récupérer la fréquence f9960 à partir de la valeur moyenne du signal REF démodulé.

Le signal VAR est ensuite filtré grâce à la fonction filtrage passe bas et sa moyenne est récupérée.

Les fonctions de recherche de zéros sont ensuite appliquées successivement aux deux signaux traités.

Pour prévoir le déphasage dû au filtre de Hilbert on récupère ensuite la fréquence du signal REF grâce à la fonction « freq ».

Un algorithme permet ensuite de mesurer l'avance de phase du VAR à partir des deux tableaux de zéros. Ce déphasage brut est ensuite corrigé du déphasage dû au filtre de Hilbert et à la dérivation.

La phase ainsi obtenue est translatée dans l'intervalle [0,360] puis mémorisée dans le buffer tournant.

Enfin un algorithme permet de moyenner les valeurs de phases contenues dans le buffer tournant en excluant 4 valeurs extrêmes (les deux valeurs maximales et minimales).

Un algorithme spécifique est employé si les valeurs maximums et minimums sont éloignées d'au moins 355°. En effet ,dans ce cas la phase doit être proche de 0° ou 360° et chaque phase mémorisée doit être ramenée au voisinage de 0 pour éviter l'effet de la discontinuité 0° -360°

Il suffit pour cela de soustraire 360 aux phases supérieures à 355 avant de recalculer la moyenne.

4. La procédure d'interruption

Après l'acquisition de la valeur d'entrée, celle-ci reçoit un bit de signe et est convertie sur les 8 bits de poids faible pour éviter de dépasser par la suite la valeur maximum quantifiable.

Cette valeur est ensuite dupliquée pour pouvoir faire le traitement du VAR dans la suite du main.

Le REF est ensuite appliqué au filtre de Hilbert qui comporte deux composantes : une réelle et une imaginaire. Celles-ci sont obtenues en prenant les parties réelle et imaginaire du filtre donné par Matlab.

Après l'utilisation de la fonction atan2f sur le couple (partie Imaginaire,partie Réelle) , l'algorithme de dépliage-dérivation puis le filtrage passe-bas sont ensuite appliqués.

Enfin la valeur REF est sommée à une variable spécifique afin de calculer la moyenne sur 1000 échantillons.

5. Les opérations de filtrage

En vue du filtrage les coefficients des filtres sont importés au début du programme. On importe ainsi 3 séries de coefficients : les coefficients du passe bas et les deux séries de coefficients du filtre de Hilbert (parties réelle et imaginaire)

La fonction FIR nécessite aussi l'utilisation de tampons servant à mémoriser les valeurs d'entrée précédentes. Ces tampons ont pour premier élément un entier indiquant la position de la valeur d'entrée courante ainsi que 101 autres valeurs dans le cas de notre filtre d'ordre 100. Ils ne demandent pas à être remis à zéro car les anciennes valeurs n'ont d'influence que sur la phase transitoire qui n'est pas prise en compte pour le calcul du déphasage.

La fonction FIR demande aussi en entrée le nombre de coefficients du filtre utilisé (101 dans notre cas)

6. Mode d'emploi

Une fois le programme lancé et la phase à mesurer sélectionnée, l'utilisateur doit être sûr que le buffer tournant est entièrement rempli. Pour cela un clignotement de LED est commandé si l'indice du buffer est inférieur à 4. Ainsi si l'utilisateur modifie la phase entre deux clignotements, il devra attendre deux séries de clignotements et s'il la modifie pendant un clignotement, il attendra la fin du clignotement suivant.

Notre programme final nécessite 8 secondes pour effectuer un cycle du buffer tournant.

Une tension continue représentant la phase moyenne est aussi injectée en sortie afin d'être visualisée sur oscilloscope. Cela permet d'observer une valeur approchée de la phase.

Pour obtenir la phase de façon précise, on devra stopper momentanément le programme (F4) et accéder à la mémoire du DSP.

V. CONCLUSION

Le programme et le matériel utilisés nous ont permis de répondre au cahier des charges (précision de 0.1° sur le déphasage) après 8 secondes d'acquisitions.

Un matériel plus adéquat aurait cependant permis d'améliorer ces résultats. En effet nous n'avons utilisé qu'une partie de la plage de quantification du DSP ; ce qui revient à réduire le taux de quantification. Or les résultats obtenus sur Matlab mettent en évidence ses effets : la précision est dégradée de façon significative.

L'idéal serait d'utiliser un DSP fonctionnant sur la plage [0.5V,3.5V] avec une quantification sur 16 bits. Par contre la fréquence d'échantillonnage à 30 KHz semble suffisante ; son augmentation sous Matlab n'améliore pas sensiblement les résultats.

VI. ANNEXES

1. Programme C

```
/*************************************************************************/
/* Module : MANIP_2.C                                                    */
/* Auteur : LUCAS_THOMAS, ENAC L01                                       */
/* Responsable de projet : M  J.M. LOUIS                                 */
/* Date   : JUIN 2003                                                    */
/*************************************************************************/

#define fe 30000
#define ordreb 100
#define ordreh 100
#define pi 3.14159265358979
#define nb_ech 11500
#define nb_zer 16
#define nb_saisies 20

#include <21020.h>
#include <signal.h>
#include <filters.h>
#include <macros.h>
#include <math.h>
#include <stats.h>
#include <fir9960r.h>
#include <fir9960i.h>
#include <fir30.h>


/*******************DEFINITION DES ACCES CONVERTISSEURS *************/

DEF_PORT(in_port_analogique_vers_numerique,unsigned int,adc_b,dm);
DEF_PORT(out_port_b,unsigned int,dac_b,dm);
DEF_PORT(out_port_a,unsigned int,dac_a,dm);

/************************DEFINITION DES VARIABLES*********************/

float sample,sample_outr,sample_outi,moy_ref,moy_var;

int valeur_en_entree;
int valeur_en_sortie;

int sortie;
int NB_IT_Echantillon;
int i0=ordreb+ordreh+20; /*pour eliminer phases transitoires */
```

```
float tampon1[ordreh+2],tampon2[ordreh+2],tampon3[ordreb+2],tampon4[ordreb+2];
float ref[nb_ech],Svar[nb_ech];
float f30,f9960;
float phase_M;
float Buffer[nb_saisies];


/************************Fonction frequence mesuree**********************/
float freq(float *t)
{
int i;
float m=0;
float buf[nb_zer-2];
for (i=0 ; i < nb_zer-2 ; i++)
                    {
                    buf[i]=t[i+1]-t[i];
                    m+=buf[i];
                    }
return((nb_zer-2)/(m*2));
}

/***********************Fonction moyenne*********************************/
float moyenne(float *buf,int deb,int n)
{
int i;
float m;
m=0;
for (i=deb;i<deb+n;i++)
                    {
                    m+=buf[i];
                    }
m=m/n;
return m;
}

/*****************************Fonction filtre PB30************************/
void filtrepb(float *e,float *tmp)
{
int i;
for(i=ordreh+10;i<nb_ech;i++)
          {
                    e[i]= fir(e[i],an_filtre30,tmp,ordreb+1);
          }
}
```

```
/**************************Fonction de recherche des zéros******************/
void rech_zero(float *e,float *te,float moy)
{
float A,B;
int n=0,i=0;

while (n<nb_zer)
        {
        if ((e[i0+i+1]-moy)*(e[i0+i]-moy)<0)
                {
                A=(i-1)*(e[i0+i-1]-moy)+i*(e[i0+i]-moy)+(i+1)*(e[i0+i+1]-
moy)+(i+2)*(e[i0+i+2]-moy);
                B=e[i0+i-1]+e[i0+i]+e[i0+i+1]+e[i0+i+2]-4*moy;
            te[n]=   ((4*i+2)*A-B*(4*i*i+4*i+6)) / (4*A-B*(4*i+2))  /fe;
                /*te[n]=(i*e[i0+i+1]-e[i0+i]*(i+1))/(e[i0+i+1]-e[i0+i])/fe;*/
                n=n+1;
                i=i+1+fe/60-50;
                }
        if (e[i0+i]==moy)
                {
                te[n]=i/fe;
                n=n+1;
                i=i+fe/60-50;
                }
        i=i+1;
        }
}

/****************Procedure d'echantillonage par interruption***************/

void it_d_echantillonnage(int interrupt_Number)
{

sample=1;

valeur_en_entree= in_port_analogique_vers_numerique;
/* Acquisition de la valeur sur 8 bits*/

in_port_analogique_vers_numerique= 0; /* Ordre de nouvelle conversion */

valeur_en_entree= valeur_en_entree^0x80000000;  /* Bit de signe */
valeur_en_entree= valeur_en_entree >> 24;    /* Decalage a droite */

sample= (float) valeur_en_entree;    /* et conversion au format ad hoc */

ref[NB_IT_Echantillon]= sample;

Svar[NB_IT_Echantillon]=ref[NB_IT_Echantillon];
```

```
/*-------------------------filtrage de hilbert du ref-------------------------*/

sample_outr = fir(ref[NB_IT_Echantillon],an_filtre9960r,tampon1,ordreh+1);
sample_outi = fir(ref[NB_IT_Echantillon],an_filtre9960i,tampon2,ordreh+1);
ref[NB_IT_Echantillon]= -atan2f(sample_outi,sample_outr);

/*-------------------------depliage derivation du ref-----------------------*/

if(NB_IT_Echantillon>=ordreh+11)
{
 if( (ref[NB_IT_Echantillon]-ref[NB_IT_Echantillon-1]) <(-pi))
    ref[NB_IT_Echantillon-1]=ref[NB_IT_Echantillon]-ref[NB_IT_Echantillon-1]+2*pi;

  else
 {
  if ((ref[NB_IT_Echantillon]-ref[NB_IT_Echantillon-1])>pi)
     ref[NB_IT_Echantillon-1]=ref[NB_IT_Echantillon]-ref[NB_IT_Echantillon-1]-2*pi;
  else
     ref[NB_IT_Echantillon-1]=ref[NB_IT_Echantillon]-ref[NB_IT_Echantillon-1];
 }
}

/*--------------------------filtrage passe bas du ref----------------------*/

ref[NB_IT_Echantillon-1]=fir(ref[NB_IT_Echantillon-1],an_filtre30,tampon3,ordreb+1);


/*-------------------------calcul de la moyenne du ref----------------------*/

if(( (NB_IT_Echantillon-1)>=i0)&&( (NB_IT_Echantillon-1)<i0+fe/30))
          moy_ref+=ref[NB_IT_Echantillon-1];


NB_IT_Echantillon++;
}

/*****************************PROGRAMME PRINCIPAL*******************/
int main(void)
{
int i;
int k=0;
int d=0;
float Max1,Min1,Max2,Min2,phase;
float tref[nb_zer],tvar[nb_zer];

in_port_analogique_vers_numerique= 0;  /* Ordre initial de conversion */
interrupt(SIG_TMZ0, it_d_echantillonnage);
/* Definition et armement de l'IT d'echantillonnage */
sample=1;
```

```
timer_set(1100, 1100);
/*choix de la frequence d'echantillonage = 30 KHZ */
                        /* Te = 1/fe = a / 33000000 , a etant le parametre */


                        /* de timer_set()  ,  pour carte a quarzt a 33 Mhz */

/*initialisation des LEDS*/
set_flag(SET_FLAG1,SET_FLAG);
set_flag(SET_FLAG2,SET_FLAG);
set_flag(SET_FLAG3,SET_FLAG);

while(1)
{
timer_on();

moy_ref=0;
moy_var=0;

/*******************************ACQUISITION*****************************/
sortie=0;
NB_IT_Echantillon=0;

while (NB_IT_Echantillon<nb_ech)
  {
   idle();       /*  Blocage et Attente IT !!  */
  }

timer_off();

/*sortie de phase moyenne sur l'oscilloscope*/
valeur_en_sortie = (int) ((phase_M*2/3)-120);
valeur_en_sortie = valeur_en_sortie << 24;    /* Decalage a droite */
valeur_en_sortie = valeur_en_sortie^0x80000000;  /* Bit de signe */
out_port_b = valeur_en_sortie;

/* commande de clignotement*/
if (k<4)
          {set_flag(SET_FLAG1,TGL_FLAG);
          set_flag(SET_FLAG2,TGL_FLAG);
          set_flag(SET_FLAG3,TGL_FLAG);
          }

moy_ref=moy_ref*30/fe; /*% par fe/30(nbech pour une periode du 30 Hz)*/


/************************RECUPERATION DE W9960************************/
f9960=fe*moy_ref/2/pi;
```

```
/*****************************FILTRAGE DU VAR*****************************/
filtrepb(Svar,tampon4);

/****************CALCUL DE LA VALEUR MOYENNE (VAR)********* ********/
moy_var=moyenne(Svar,i0,1000);  /*moyenne sur une periode de 30 Hz;*/

/******************CONSTRUCTION DES TABLEAUX DE ZEROS*************/
rech_zero(ref,tref,moy_ref);
rech_zero(Svar,tvar,moy_var);

/****************CALCUL DES FREQUENCES DU REF ET DU VAR*************/
f30=freq(tref);

/*********************DETERMINATION DU DEPHASAGE BRUT***************/


if((ref[i0]-moy_ref)*(Svar[i0]-moy_var)>0)
        {
        for(i=0;i<nb_zer;i++)
        tref[i]=f30*(tref[i]-tvar[i])*2*180;
        phase=moyenne(tref,0,nb_zer);
        }
else
        {
        for(i=0;i<nb_zer-2;i++)
        tref[i]=f30*(tref[i+1]-tvar[i])*2*180;
        phase=moyenne(tref,0,nb_zer-2) ;
        }

/***********CORRECTION DU DEPHASAGE DU AU FILTRE DE HILBERT********/
phase=phase+(-180*f30*ordreh/fe);

/************CORRECTION DU DEPHASAGE DU A LA DERIVATION***********/
phase=phase+1*f30*2*180/(2*fe);

/**********************SAISIE DANS LE BUFFER*************************/
Buffer[k]=fmodf(phase,360);
if(Buffer[k]<0)
        {
        Buffer[k]+=360;
        }

if(k==nb_saisies-1)
        {
        k=0;
        d=1;
        }
else
        k++;
```

```
if((d==1)||(k-1>3)) /*d=1 si buffer plein */
{
phase_M=0;
Min1=1000;
Max1=-1000;
Min2=1000;
Max2=-1000;
for(i=0;i<nb_saisies*d+(1-d)*(k+1);i++)
        {
        if(Buffer[i]<Min1)
                {
                Min2=Min1;
                Min1=Buffer[i];
                }
        else
                {
                if(Buffer[i]<Min2)
                        Min2=Buffer[i];
                }

        if(Buffer[i]>Max1)
                {
                Max2=Max1;
                Max1=Buffer[i];
                }
        else
                {
                if(Buffer[i]>Max2)
                                Max2=Buffer[i];
                }
        phase_M+=Buffer[i];
        }

phase_M-=Max1+Min1+Max2+Min2;
phase_M=phase_M/(nb_saisies*d+(1-d)*(k+1)-4);


if((Max1-Min1>355)) /*Max1-Min1>355 si phase proche de 0 ou 360°*/
        {
        phase_M=0;
        Min1=1000;
        Max1=-1000;
        Min2=1000;
        Max2=-1000;
        for(i=0;i<nb_saisies*d+(1-d)*(k+1);i++)
                {
                if(Buffer[i]>355)
                                Buffer[i]-=360;
```

```
                if(Buffer[i]<Min1)
                                {
                                Min2=Min1;
                                Min1=Buffer[i];
                                }
                else
                                {
                                if(Buffer[i]<Min2)
                                        Min2=Buffer[i];
                                }

                if(Buffer[i]>Max1)
                                {
                                Max2=Max1;
                                Max1=Buffer[i];
                                }
                else
                                {
                                if(Buffer[i]>Max2)
                                        Max2=Buffer[i];
                                }
                phase_M+=Buffer[i];
                }
        phase_M-=Max1+Min1+Max2+Min2;
        phase_M=phase_M/(nb_saisies*d+(1-d)*(k+1)-4);
        }
}

}
}
```

2. Programme Matlab

```
clc;clear all;close all;

%*******************************      paramètres    **********************
fcb=90;
fb=30;
fh=9960;
q=64;
fe=30000;
ordreb=100;
ordreh=100;
%*************************************************************************
```

```
% signal

t=[0:1/fe:20/fb];
e=floor((q-1)*(cos(2*pi*fh*t+16*sin(2*pi*fb*t))+cos(2*pi*fb*t+60*pi/180)));

%e=load('120_3.plt');

%Hilbert

B0=fir1(ordreh,480/(fe/2));

for l=1:1:ordreh+1;
  Bh(l)=B0(l)*exp(i*(l-1)*2*pi*9960/fe);
end


s=filter(Bh,1,e);
s=s(ordreh+10:length(s));

sr=real(s);
si=imag(s);
s1=atan2(si,sr);


l=length(s);
for k=2:1:l

   if s1(k)-s1(k-1)<-pi
      s1(k-1)=s1(k)-s1(k-1)+2*pi;
  else

        if s1(k)-s1(k-1)>pi
              s1(k-1)=s1(k)-s1(k-1)-2*pi;

        else

            s1(k-1)=s1(k)-s1(k-1);
        end

    end

end

%  passe bas

B=fir1(ordreb,fcb/(fe/2));
Sref=filter(B,1,s1);
Sref=Sref(ordreb+10:length(Sref));
Sref=Sref-mean(Sref(1:1000));

%*************************************   VAR    ************************

% passe bas


Svar=filter(B,1,e);
Svar=Svar(ordreb+ordreh+20:length(Svar));
Svar=Svar-mean(Svar(1:1000));
```

```
%   **********************************   DEPHASAGE   **********************

%détection des zéros

n=0;i=2;

while n<18;
   if Sref(i+1)*Sref(i)<0;
      A=(i-1)*Sref(i-1)+i*Sref(i)+(i+1)*Sref(i+1)+(i+2)*Sref(i+2);
      B=Sref(i-1)+Sref(i)+Sref(i+1)+Sref(i+2);
      t1(n+1)=(   (  ((4*i+2)*A-B*(4*i^2+4*i+6)) / (4*A-B*(4*i+2))  )
/fe);
      n=n+1;
      i=i+round(fe/fb/2)-50;
    end
   if Sref(i)==0;
      t1(n+1)=(i/fe);
      n=n+1;
      i=i+round(fe/fb/2)-50;
   end
   i=i+1;
end


m=0;i=2;
while m<18
   if Svar(i+1)*Svar(i)<0
      A=(i-1)*Svar(i-1)+i*Svar(i)+(i+1)*Svar(i+1)+(i+2)*Svar(i+2);
      B=Svar(i-1)+Svar(i)+Svar(i+1)+Svar(i+2);
      t2(m+1)=(   (  ((4*i+2)*A-B*(4*i^2+4*i+6)) / (4*A-B*(4*i+2))  )
/fe);
      m=m+1;
      i=i+round(fe/fb/2)-50;
  end

   if Svar(i)==0
      t2(m+1)=(i/fe);
      m=m+1;
      i=i+round(fe/fb/2)-50;
  end

   i=i+1;
end

%mesure de la fréquence

for r=1:16
   dt(r)=t1(r+1)-t1(r);
end
frequence=1  /  (  mean(dt)*2 )

fi1=t1*frequence*2*pi;
fi2=t2*frequence*2*pi;

%correction de la phase apportée par le filtre de Hilbert
fi2=fi2-pi*(-frequence)*ordreh/fe;

%..et par la dérivée
fi2=fi2+(1/(2*fe))*frequence*2*pi;
```

```
%mesure du déphasage
if Sref(2)*Svar(2)>0
   dfi=fi1-fi2;
   mu=mean(dfi);
else
  for r=1:16;
      dfi(r)=fi1(r+1)-fi2(r);
      mu=mean(dfi);
  end
end

%positionnement dans -pi:pi
while abs(mu)>pi
      mu=mu-sign(mu)*2*pi;
end

%affichage du déphasage en degrés

dephasage=mu*180/pi
```