TP SYNCHRONISATION

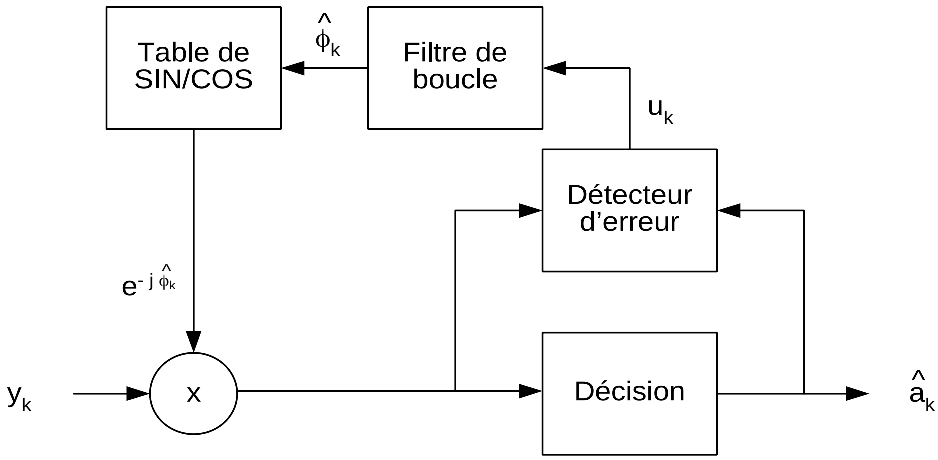

Fig. 1: boucle à verrouillage de phase

On transmet N symboles a_k au rythme 1/T sur un canal à bruit additif blanc gaussien (BABG). Les symboles complexes a_k sont indépendants, équiprobables et choisis dans la constellation $C=\{\sqrt{E_s}\,e^{j\pi/4}, \sqrt{E_s}\,e^{j3\pi/4}, \sqrt{E_s}\,e^{j5\pi/4}, \sqrt{E_s}\,e^{j7\pi/4}\}$, où E_s désigne l'énergie moyenne par symbole. Après démodulation, la sortie du canal peut alors s'écrire sous la forme :

$$y(t) = \exp[j(\phi+2\pi f_D t)] \sum_{k=0}^{N-1} a_k\, g(t-kT-\varepsilon T)+n(t). \qquad (1)$$

Les paramètres ϕ, f_D et ε représentent respectivement le déphasage, le décalage fréquentiel et le retard fractionnaire introduits par le canal.

La forme d'onde g(t) supposée triangulaire s'écrit

$$g(t)=1-|t|/T \text{ si } -T< t <T \text{ et } g(t)=0 \text{ sinon.}$$

Le bruit complexe $n(t) = n_I(t)+ j\, n_Q(t)$ est tel que

$$n_I(t) \sim N(0,N_0/2) \text{ et } n_Q(t) \sim N(0,N_0/2) \text{ avec } n_I(t) \text{ et } n_Q(t) \text{ indépendants.}$$

La sortie du canal est échantillonnée au rythme de un échantillon par symbole. On obtient ainsi le vecteur d'observations complexes $Y=[y_0, y_1, \ldots, y_{N-1}]$.

1. Ecrire un sous-programme Python `def gendata(N,Es) :` permettant de générer un vecteur data contenant la suite des N symboles aléatoires a_k, k=0,…,N-1.

2. Ecrire un sous-programme Python `def RX(t,T,data,N0,phi,fD,epsilon) :` permettant de simuler un échantillon y_t de la sortie du canal à un instant t, pour un déphasage phi, un décalage fréquentiel f_D et un retard fractionnaire epsilon quelconques.

3. A partir du sous-programme de la question 2, générer un vecteur de N observations complexes $Y=[y_0, y_1, \ldots, y_{N-1}]$ avec

   $y_k = y(t)\big|_{t=kT-T/10}$, $k=0,1,\ldots,N-1$.

   Les paramètres utilisés seront les suivants : N=1000, $E_s/N_0 = 20$ dB, $\phi=\pi/8$, $f_DT=0$, $\varepsilon=-0.4$.
   Tracer ces échantillons dans le plan complexe avec la commande `plt.plot(np.real(Y),np.imag(Y),'+')`. Expliquer le résultat obtenu.

4. Reprendre la question 3 en remplaçant les instants d'échantillonnage $t= kT-T/10$, $k=0,1,\ldots,N-1$ par les instants d'échantillonnage optimaux. Dans toute la suite du TP, on travaillera avec les instants d'échantillonnage optimaux supposés connus du récepteur.

5. Ecrire un sous-programme Python correspondant à la boucle à verrouillage de phase de la Fig.1 en prenant comme détecteur d'erreur de phase résiduelle

   $$u_k=\mathrm{Im}[\hat{a}_k^* y_k \exp(-j\hat{\phi}_{k-1})]/E_s$$

   et comme filtre de boucle un intégrateur du premier ordre de gain γ.
   On aura donc un sous-programme de la forme,
   def PhaseEstimation(Y,gamma, phi_chap_0,Es) :
   où Y représente le vecteur de N échantillons en sortie du canal, Es l'énergie moyenne par symbole, phi_chap_0 la valeur initiale $\hat{\phi}_0$ et gamma le gain de boucle. Les sorties phi_chap,a_chap sont les vecteurs de longueur N contenant les valeurs de $\hat{a}_k$ et $\hat{\phi}_k$, $k=0,1,\ldots,N-1$.

6. Tester le programme de la question 5 avec comme paramètres N=1000, $E_s/N_0 = 20$ dB, $\phi=\pi/8$, $f_DT=0$, $\varepsilon=-0.4$, $\hat{\phi}_0=0$, $\gamma=0.01$. L'estimateur de phase converge-t-il vers la bonne valeur ? Calculer le taux d'erreur symbole.

7. Reprendre la question 6 en remplaçant γ par 0.1. Commentaire.

8. Reprendre la question 6 en remplaçant ϕ par $3\pi/8$. Expliquez le résultat observé à partir de la courbe en S du détecteur obtenue en cours.

9. Reprendre la question 6 en remplaçant f_DT par 10^{-4}. Commenter l'allure du graphe dans le plan complexe obtenu par la commande, `plt.plot(np.real(Y),np.imag(Y),'+')`. Mesurer le biais d'estimation de la phase et expliquer son origine. Sa valeur est-elle conforme à la valeur attendue ?

10. Reprendre la question 6 et estimer l'erreur quadratique moyenne après convergence entre la phase et la phase estimée. Obtient-on la valeur attendue ?