Synchronisation de phase - Boucle à remodulation souple

Stéphanie Bay[1], Christophe Vanstraceele[1], Benoit Geller[1,3], Jean-Pierre Barbot[1], Jean-Marc Brossier[2] *

[1]Laboratoire SATIE / UMR 8029
ENS Cachan 61, avenue du Président Wilson 94235 Cachan Cedex, France

[2]Laboratoire LIS
ENSIEG, 961, av. de la houille blanche, Domaine Universitaire BP 46, 38402 Saint Martin d'Hères

[3]Université Paris 12
av. du Général de Gaulle, 94010 Créteil Cedex

bay@satie.ens-cachan.fr, vanstraceele@satie.ens-cachan.fr, geller@satie.ens-cachan.fr,
barbot@satie.ens-cachan.fr, jean-marc.brossier@lis.inpg.fr

Résumé – Les performances des systèmes de transmission numérique dépendent de nombreux facteurs. Parmi ceux-ci, l'étape de synchronisation est déterminante pour la qualité de réception de l'information numérique transmise. En effet, les améliorations apportées par les codes correcteurs d'erreurs et les turbo-codes en particulier sont fortement tributaires de l'étape de synchronisation. Nous présentons dans cet article un algorithme permettant d'estimer et de compenser la phase de la porteuse du signal reçu en exploitant l'information souple fournie par tout turbo-décodeur. La méthode proposée permet de faire une mise à jour de la phase à chaque symbole grâce à une boucle adaptative utilisant les log-vraisemblances de chaque bit reçu fournies par un turbo-décodeur. De plus, une boucle "aller-retour" permet d'améliorer sensiblement le comportement de cet estimateur de phase et ainsi d'obtenir des performances, en terme d'Erreur Quadratique Moyenne, proches de la boucle à symboles connus.

Abstract – The phase estimation is a really important step in digital communication systems. Without a good synchronization, using error correction codes and turbo-codes is suboptimum. This paper describes a new synchronizing algorithm which can update the phase of each received symbol. The estimated phase is calculated with a Soft Decision Feedback Loop. This loop uses the Log-Likelihood Ratio (LLR) provided by the turbo-decoder's soft output. A forward-backward loop improves the phase estimation. This algorithm can be applied on every QAM's constellations and good performance is achieved till near Shannon's limit SNRs.

1 Introduction

Le turbo-décodage permet des performances proches de la capacité de Shannon à faible rapport signal à bruit [1][4]. Mais ces performances supposent une synchronisation parfaite : la moindre erreur de phase entraîne une dégradation importante du taux d'erreur binaire comme l'illustre la figure 1. Ainsi, il est indispensable d'estimer et de compenser correctement la phase de la porteuse du signal reçu. Nous proposons donc un algorithme exploitant les informations souples provenant d'un turbo-décodeur. Ces informations sont alors insérées dans une boucle adaptative qui agit ensuite sur la phase du signal reçu.

2 Présentation du modèle

Le système considéré dans cet article est un système de communication numérique utilisant un code correcteur produit et des constellations QAM (Quadrature Amplitude Modulation) de taille fixée variable entre 2-QAM et 1024-QAM. Le canal de transmission considéré introduit une erreur sur la phase de la porteuse et ajoute un Bruit Additif Blanc Gaussien (BABG). Les symboles émis subissent donc une rotation due à l'erreur de phase et sont perturbés par le BABG :

*Cette étude a été partiellement financée par le contrat européen Newcom n°507325

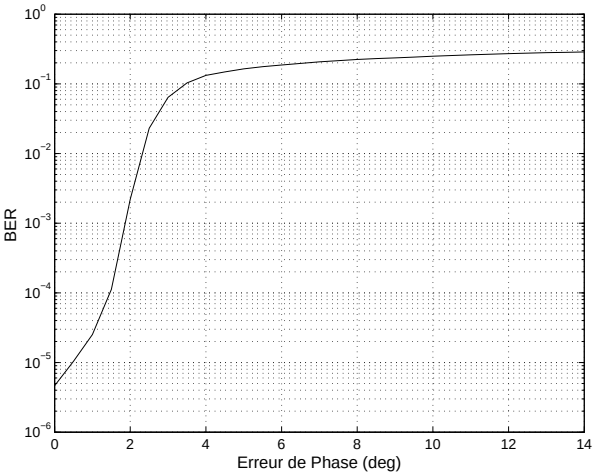

Fig. 1 – Conséquence d'une erreur de phase sur le TEB (Taux d'Erreur Binaire) (BCH(32,26,4)^2 sur 1024-QAM avec Eb/No=17.5dB)

$$y_k = a_k e^{i\varphi_k} + b_k \tag{1}$$

pour $k = 1...K$, avec $a_k \in \{Q_1...Q_M\}$ symboles indépendants d'une constellation M-QAM où $M = 2^N$, b_k le terme de bruit additif de moyenne nulle et de variance σ_b^2.

L'erreur de phase φ_k quant à elle, est supposée avoir une évolution brownienne s'ajoutant à une dérive linéaire

[2] :

$$\varphi_{k+1} = \varphi_k + \Delta\varphi + w_k \tag{2}$$

où $\Delta\varphi$ est une dérive linéaire due à un décalage de fréquence et w_k est un bruit gaussien de moyenne nulle de variance σ_Φ^2 portant sur la phase du signal (gigue).

A la réception, nous utilisons pour illustrer le fonctionnement de cette boucle de phase un décodeur de Pyndiah [6], applicable à n'importe quel code produit construit à partir de codes blocs linéaires. Ce décodeur a l'avantage de converger plus rapidement que les autres turbo-décodeurs. Sa sortie souple est une estimation de la log-vraisemblance calculée grâce aux décisions données par le décodeur de Chase [3]. Concernant la synchronisation, [7] propose un algorithme de synchronisation itératif exploitant la sortie du décodeur de Pyndiah. Cet algorithme de synchronisation nécessite une phase constante sur un mot de code. L'algorithme que nous proposons dans cet article permet de suivre les variations rapides de phase à l'intérieur même d'un mot de code.

Les observations forment le vecteur $Y = (y_1,...,y_K)^T$. On peut exprimer la vraisemblance des observations :

$$P\left(Y|\theta, a_1,..., a_K\right) = \prod_{k=1}^{K} \frac{1}{\pi\sigma_b^2} \exp\left(-\frac{\left|y_k - a_k e^{i\theta}\right|^2}{\sigma_b^2}\right) \tag{3}$$

où θ est le paramètre d'intérêt (ici $\theta = \varphi$ phase que l'on souhaite estimer). Maintenant, en tenant compte de l'information a priori sur les symboles a_k inconnus, avec $a_k \in \{Q_1,...,Q_M\}$, on obtient :

$$P\left(Y|\theta\right) = \sum_{k=1...K} P\left(Y|\theta, a_1,..., a_K\right) P\left(a_1,..., a_k,..., a_K\right) \tag{4}$$

En considérant que les bits sont indépendants, on peut exprimer la probabilité d'un symbole en fonction de la probabilité de chaque bit :

$$P\left(a_k = Q_m\right) = \prod_{n=1}^{N} P\left(b_n^k = q_n^m\right) \tag{5}$$

où $b_n^k \in \{-1, +1\}$ représente le n^{ieme} bit du k^{ieme} symbole QAM. La probabilité d'un bit s'exprime alors en fonction de sa log-vraisemblance :

$$P\left(b_n^k = q_n^m\right) = \frac{e^{\frac{q_n^m L_n^k}{2}}}{2\cosh\left(\frac{L_n^k}{2}\right)} \tag{6}$$

où L_n^k est la log-vraisemblance sur le n^{ieme} bit du k^{ieme} symbole QAM émis. On en déduit alors la probabilité d'un symbole :

$$P\left(a_k = Q_m\right) = \left[\prod_{n=1}^{N} \frac{1}{2\cosh\left(\frac{L_n^k}{2}\right)}\right] \times \exp\left(\frac{1}{2}\sum_{n=1}^{N} q_n^m L_n^k\right) \tag{7}$$

En groupant les termes en exponentielle, l'équation (7) devient :

$$P\left(Y\ |\theta\right) = \left(\frac{1}{\pi\sigma_b^2}\right)^K \left[\prod_{k=1}^{K}\left(\prod_{n=1}^{N} \frac{1}{2\cosh\left(\frac{L_n^k}{2}\right)}\right)\right] \times \sum_{m=1}^{M} W_m\left(y_k, L_k, \theta\right) \tag{8}$$

où

$$W_m\left(y_k, L^k, \theta\right) = \exp\left(\frac{1}{2}\sum_{n=1}^{N} q_n^m L_n^k - \frac{\left|y_k - e^{i\theta}Q_m\right|^2}{\sigma_b^2}\right) \tag{9}$$

avec $L^k = \left(L_1^k,..., L_n^k,..., L_N^k\right)$.

Seul le terme $W_m\left(y_k, L^k, \theta\right)$ dépend de θ. La dérivée de la log-vraisemblance est donc la suivante :

$$\frac{\partial}{\partial\theta}\log P\left(Y\ |\theta\right) = \sum_{k=1}^{K} \frac{\partial}{\partial\theta}\log \sum_{m=1}^{M} W_m\left(y_k, L^k, \theta\right) = \frac{2}{\pi\sigma_b^2}\sum_{k=1}^{K} \frac{\sum_{m=1}^{M} \mathrm{Im}\left(y_k e^{-i\theta}\overline{Q_m}\right) W_m\left(y_k, L^k, \theta\right)}{\sum_{m=1}^{M} W_m\left(y_k, L^k, \theta\right)} \tag{10}$$

L'estimateur du Maximum de Vraisemblance est obtenu en annulant cette expression. Mais la recherche d'une telle solution est irréalisable. On est alors amené à mettre en place une boucle adaptative qui corrige θ à chaque symbole reçu.

3 Boucle à remodulation souple

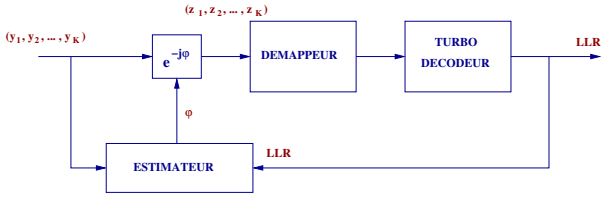

FIG. 2 – Boucle de phase à remodulation souple

On a introduit l'information "a priori" sur les symboles a_k fournie par la sortie du turbo-décodeur [6]. L'estimateur du Maximum de Vraisemblance étant trop complexe, on construit la boucle adaptative de la figure 2 qui corrige la phase à chaque nouveau symbole reçu. L'implantation de l'algorithme, utilisable pour n'importe quel type de modulation QAM, se trouve alors allégée :

$$\hat{\varphi}_k = \hat{\varphi}_{k-1} + \gamma \frac{\sum_{m=1}^{M} Im\left(y_k\overline{Q_m}e^{-i\hat{\varphi}_{k-1}}\right) W_m\left(y_k, L^k, \hat{\varphi}_{k-1}\right)}{\sum_{m=1}^{M} W_m\left(y_k, L^k, \hat{\varphi}_{k-1}\right)} \tag{11}$$

Dans cette boucle, il est important de bien doser l'information souple réinjectée. W_m (équation 11), peut alors être interprété comme un poids assigné à chaque symbole possible Q_m appartenant à la constellation considérée :

$$W_m\left(y_k, L^k, \theta\right) = exp\left(K_{DFL}\sum_{n=1}^{N} q_n^m L_n^k - \frac{\left|y_k - e^{i\theta}Q_m\right|^2}{\sigma_b^2}\right) \quad (12)$$

où K_{DFL} est le coefficient de pondération des LLR (Log Likelihood Ratio) qui doit être optimisé en fonction des conditions de transmission. Ce poids dépend à la fois de la distance de Q_m vis à vis du symbole reçu mais aussi des log-vraisemblances des bits du symbole reçu. Ainsi, une forte log-vraisemblance et/ou une distance très faible par rapport au symbole reçu augmenteront le poids de ce symbole dans le calcul. Par conséquent la boucle tendra à calculer une nouvelle phase $\hat{\varphi}_k$ qui permettra de se rapprocher de ce symbole. A l'inverse, une faible log-vraisemblance associée à une grande distance par rapport au symbole reçu diminueront l'importance d'un symbole QAM dans le calcul.

Si pour illustrer les performances de cette boucle, nous avons choisi un turbo-décodeur de codes-produits, il est important de noter à ce stade qu'aucune hypothèse sur l'architecture du turbo-décodeur n'a été introduite, la seule donnée exploitée étant le LLR. Cette boucle peut donc également fonctionner avec un turbo-décodeur de codes convolutifs.

Cet algorithme est un algorithme de poursuite. Les simulations que nous présentons ont pour vocation d'illustrer les capacités de poursuite de cette boucle. L'algorithme a donc été initialisé avec la valeur exacte de phase, supposant par là-même que l'étape de synchronisation initiale avait été réalisée. Dans un cas concret, la boucle pourrait être initialisée par l'un des algorithmes décrit dans [5] et [7]. Lors de la première itération, l'estimation de la phase est effectuée en considérant que l'on ne possède pas d'information a priori. Dans ce cas au niveau de l'estimateur de la figure 2, LLR=0. Le bloc d'information grossièrement asservi en phase est alors appliqué à la suite de la chaîne de réception. Dès la deuxième itération, il est possible de tenir compte de l'information souple apportée par le turbo-décodeur. Plusieurs itérations peuvent être nécessaires pour la bonne convergence de l'algorithme.

4 Boucle aller-retour

Dans le cas où les paramètres ne sont pas constants, cet algorithme peut amener la présence d'un biais (erreur de traînage). Le décodeur bloc gardant en mémoire tous les symboles d'un même mot de code, une idée supplémentaire est donc de faire fonctionner la boucle dans les deux sens : sens direct et inverse (figure 3). En sens inverse, le biais apporté sera en général opposé à celui obtenu dans le sens direct, comme le montre la figure 4. En effectuant la moyenne pondérée de ces deux estimations, on montre qu'un gain de 2 à 3 dB sur l'Erreur Quadratique Moyenne (EQM) de la phase estimée est obtenu.

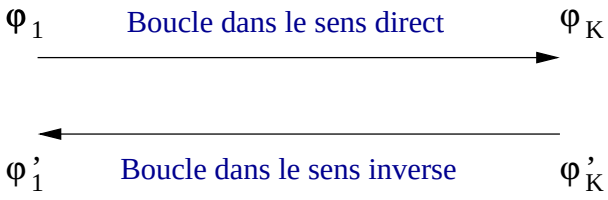

FIG. 3 – Fonctionnement aller-retour

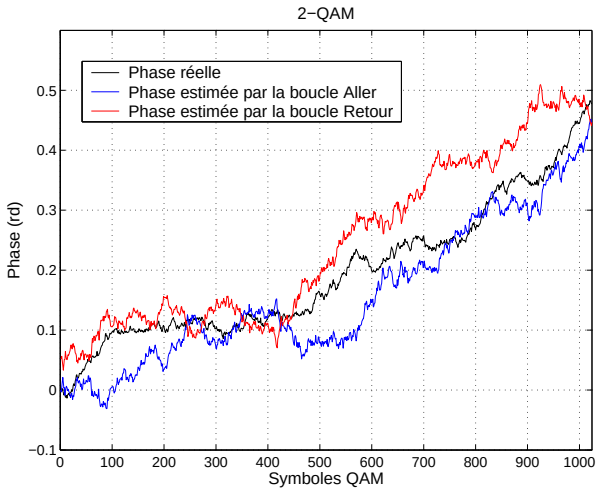

FIG. 4 – Boucles aller et retour sur une MDP-2

Une première estimation de phase est donnée en faisant tourner la boucle dans le sens direct. On obtient un premier vecteur $\hat{\varphi}_0, ..., \hat{\varphi}_K$. Ensuite, on applique le même algorithme mais maintenant en envoyant les symboles dans le sens inverse. L'initialisation de cette deuxième phase se fait en utilisant le résultat obtenu pour le dernier symbole en sens direct $\hat{\varphi}_k^{'} = \hat{\varphi}_k$. On obtient un nouveau vecteur $\hat{\varphi}_0^{'}, ..., \hat{\varphi}_K^{'}$. L'estimation finale $\hat{\varphi}_k^{"}$ pourra alors être donnée par :

$$\hat{\varphi}_k^{"} = \frac{\hat{\varphi}_k + \hat{\varphi}_k^{'}}{2} \quad (13)$$

La figure 5 nous montre l'effet de la boucle aller-retour sur l'estimation de phase pour une modulation MDP-2 (ou 2-QAM ou BPSK). La perturbation de phase est fixée pour cette simulation à $\sigma_\Phi = \pi.10^{-2}$. On peut voir que le gain en terme d'EQM approche les 3 dB pour $E_b/N_0 = 0dB$.

5 Résultats et conclusion

Dans les simulations, la perturbation sur la phase est modélisée par un gigue gaussienne (2) de variance σ_Φ^2 sans dérive linéaire. Les performances de l'estimateur de phase sont présentées en terme d'EQM. Dans le cas étudié d'une 256-QAM, pour un $E_b/N_0 = 13$dB ($TEB = 3.10^{-5}$) avec une gigue de phase $\sigma_\Phi = 6\pi.10^{-3}$rd pour le terme w_k (équation (2)), la figure 6 montre qu'il est nécessaire de trouver un compromis sur la quantité d'information réinjectée K_{DFL} dans l'itération suivante. On peut voir

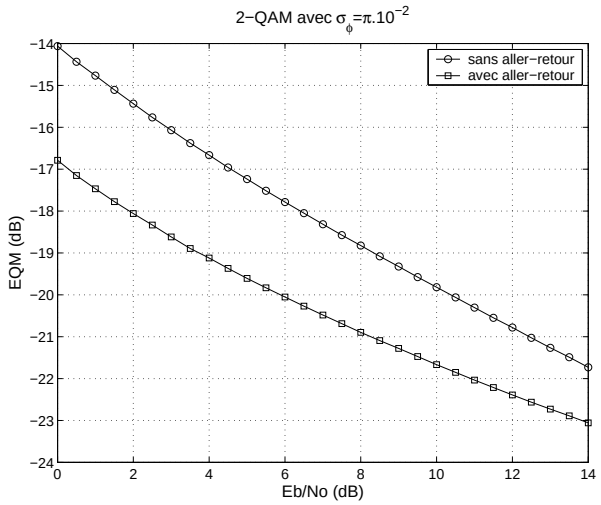

FIG. 5 – EQM pour une Boucle aller-retour sur une 2-QAM

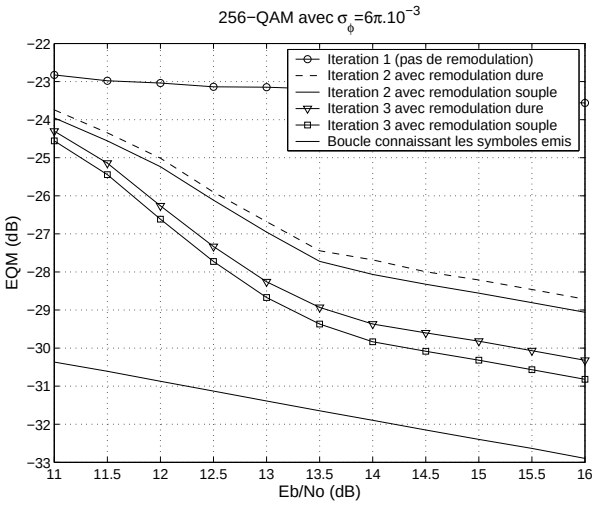

FIG. 7 – EQM pour une 256-QAM ; en fonction du rapport E_b/N_0

sur la figure 7, pour une 256-QAM, l'apport de la décision souple ($K_{DFL} = 2.10^{-1}$). On constate que la deuxième itération, qui utilise l'information provenant du turbo-décodeur, apporte un gain en terme d'EQM de 4dB (à $E_b/N_0 = 13$dB) par rapport à une boucle sans connaissance a priori. Ensuite, l'utilisation de l'information souple permet de gagner 0.4 dB par rapport à une boucle n'utilisant que les décisions dures sur les bits. La troisième itération, tirant partie de l'estimation des LLR de l'itération précédente, permet d'affiner encore l'estimation et donc de diminuer l'EQM. Dans cette configuration, l'amélioration apportée par une remodulation souple par rapport à la remodulation dure est de 0.6 dB.

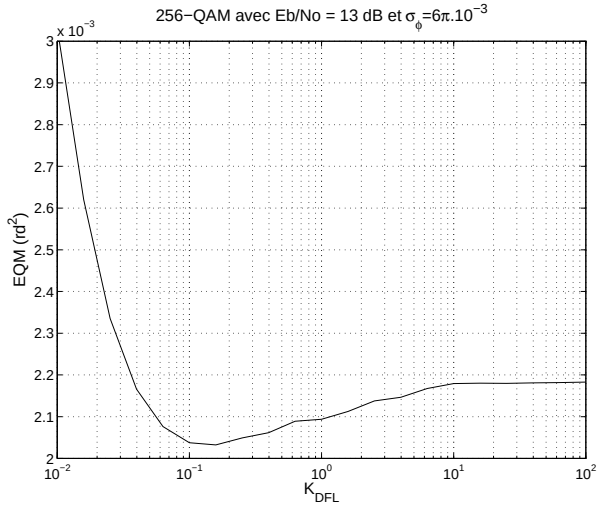

FIG. 6 – EQM en fonction du coefficient de pondération des LLR

L'algorithme proposé a l'avantage d'estimer une phase pour chaque symbole reçu. Il permet donc de poursuivre des évolutions de phase rapides à l'intérieur d'un mot de code et fonctionne pour des constellations de tailles importantes. Cette boucle à remodulation souple a été utilisée ici pour estimer et corriger la phase du signal reçu, mais elle peut également être utilisée pour estimer le gain ou tout autre paramètre d'intérêt d'un canal de transmission, y compris pour des canaux à évolutions rapides tels que certains canaux de Rayleigh.

Références

[1] C. Berrou and A. Glavieux. Near optimum error correcting coding and decoding : Turbo-codes. *IEEE Trans. Commun.*, 44(10) :1261–1271, October 1996.

[2] J.M. Brossier, P.O. Amblard, and B. Geller. Self adaptative PLL for multicarrier local loop transmission. In *Proc. Eusipco*, pages 631–634, Toulouse, September 2002.

[3] D. Chase. A class of algorithms for decoding block codes with channel measurement information. *IEEE Trans. Inform. Theory*, 18(1) :170–182, January 1972.

[4] S. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *IEEE Trans. Inform. Theory*, 42(2) :429–445, March 1996.

[5] Wrangok Oh and Kyungwhoon Cheun. Joint Decoding and Carrier Phase Recovery Algorithm for Turbo Codes. *IEEE Commun. Letters*, 5(9) :375–377, September 2001.

[6] R. Pyndiah. Near optimum decoding of product codes : Block Turbo Codes. *IEEE Trans. Commun.*, 46(8) :1003–1010, August 1998.

[7] C. Vanstraceele, B. Geller, J.P. Barbot, and J.M. Brossier. An iterative phase synchronization scheme for general QAM constellations. In *Proc. ICC'2004*, volume CT 08 Synchronization, pages 519–522, Paris, June 2004.