Influence des algorithmes d'apprentissage du modèle neuronal sur la performance de la prédistorsion des signaux FBMC-OQAM

Rakotonirina H.B.[1] , Randriamitantsoa P.A.[2], Randriamitantsoa A.A[3].

Laboratoire de Recherche en Télécommunication, Automatique, Signal et Images (LR-TASI)

Ecole Doctorale en Science et Technique de l'Ingénierie et de l'Innovation (ED – STII)

[1] dao.rakotonirina@gmail.com,[2] rpauguste@gmail.com, [3]andriau23@gmail.com

Résumé

Dans cet article, nous avons traité les signaux FBMC-OQAM (Filter Bank MultiCarrier- Offset Quadrature Amplitude Modulation) et nous avons proposé un modèle neuronal de type feedforward pour approximer la caractéristique de transfert inverse de l'amplificateur et linéariser ce denier à l'aide de la technique de prédistorsion. La performance de cette technique dépend de l'algorithme d'apprentissage du modèle neuronal. Comme critère de performance, nous avons utilisé le MSE (Mean Squared Error). Et les résultats révèlent que c'est l'algorithme de Levenberg Marquardt qui donne la meilleure performance.

Mots clés : Prédistorsion, Réseau de Neurones, Algorithme d'apprentissage, Approximation, Amplificateur, MSE

Abstract

In this article, we have processed the FBMC-OQAM (Filter Bank MultiCarrier- Offset Quadrature Amplitude Modulation) signals and we proposed a feedforward neural model to approximate the inverse transfer characteristic of the amplifier and linearize this denier using the predistortion technique. The performance of this technique depends on the learning algorithm of the neural model. As a performance criterion, we used the MSE (Mean Squared Error). And the results reveal that it is the Levenberg Marquardt algorithm that gives the best performance.

Keywords : Predistortion, Neural network, Learning algorithm, Approximation, Amplifier, MSE

1. Introduction

Grâce à sa performance, la modulation FBMC-OQAM est dorénavant l'une des modulations les plus utilisées dans les systèmes de télécommunication. Cependant, elle crée un problème de non-linéarité au niveau de l'amplificateur à cause de la forte variation d'amplitude du signal modulé. Ce problème se traduit par une dégradation du TEB à la réception ou par l'augmentation de la consommation énergétique de l'amplificateur. Afin de résoudre ce problème, nous avons proposé la prédistorsion à base de réseau de neurones de type feedforward.

Elle consiste à approximer la caractéristique de transfert inverse de l'amplificateur par un modèle neuronal et juxtaposer ce dernier avec l'amplificateur pour avoir une caractéristique de transfert linéaire (Voir figure 1)[1][2]. Cet article, nous montre l'influence des algorithmes d'apprentissage de ce modèle neuronal sur la performance de la prédistorsion d'un signal FBMC-OQAM.

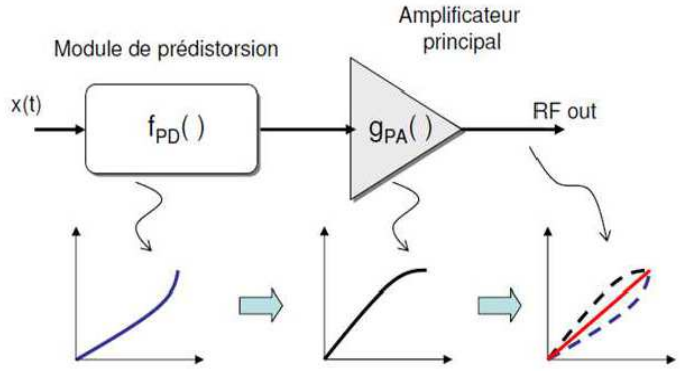

Figure 1 : *Principe de la technique de prédistorsion*

2. Réseau de neurones artificiels

2.1 Neurone formel (neurone artificiel)

La figure 2 nous montre le schéma d'un neurone formel ou neurone artificiel.

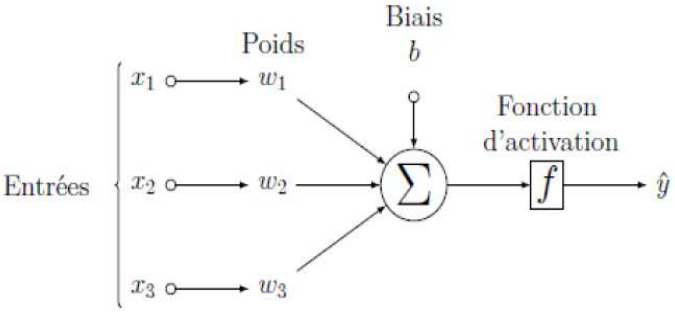

Figure 2: *Neurone artificiel*

Le neurone formel est une modélisation mathématique des principes de fonctionnement du neurone biologique. Il reçoit des variables d'entrées en provenance des autres neurones. A chacune des entrées est associé un poids w_i représentatif de la force de connexion. Les informations ainsi recueillies sont traitées par une fonction dite d'activation ou de transfert pour donner la sortie du neurone [3][4][5].

En généralisant pour n entrées, on déduit de la figure 2 l'expression de la sortie du neurone notée y.

Les entrées du neurone sont représentées par la matrice colonne x définie par l'équation 01.

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad (01)$$

La matrice w représente les poids correspondants.

$$w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} \quad (02)$$

La sortie du sommateur notée $s(x)$ a pour expression :

$$s(x) = \sum_{i=1}^{n} w_i x_i + b = w^T x + b \quad (03)$$

La sortie du neurone notée y a donc pour expression :

$$y(x) = f\big(s(x)\big) = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \quad (04)$$

Où f est la fonction d'activation du neurone et b le biais (poids synaptique correspondant à une entrée égale à +1).

2.2 Réseau de neurones de type feedforward

Un réseau de neurone artificiel est l'interconnexion de plusieurs neurones formels. Un réseau de type

feedforward est un réseau dont tous les neurones d'une couche sont reliés à tous les neurones de la couche suivante.

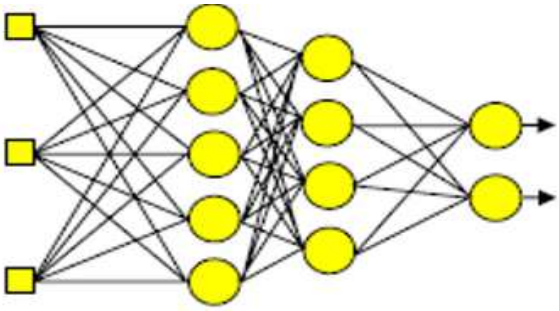

Figure 3: *Réseau de neurones de type feedforward*

2.3 Apprentissage des réseaux de neurones

Il existe deux types d'apprentissage : apprentissage supervisé et apprentissage non supervisé [4].

2.3.1 Apprentissage non supervisé

Pour le cas de l'apprentissage non supervisé, seules les valeurs d'entrée sont disponibles. Les paramètres internes du réseau sont modifiés uniquement à l'aide de ces valeurs, aucune sorties désirées ne sont prises en considération. On dit que le réseau est autodidacte.

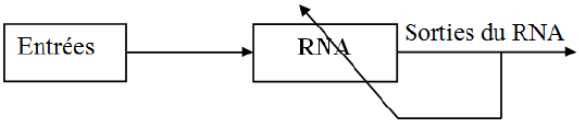

Figure 4: *Apprentissage non supervisé*

2.3.2 Apprentissage supervisé

Dans un apprentissage supervisé, on connait en avance les entrées du réseau et les sorties désirées. La phase d'apprentissage consiste donc à comparer la sortie estimée par le réseau avec la sortie désirée et mettre à jour les poids synaptiques (w_i) jusqu'à ce que le réseau donne un résultat acceptable.

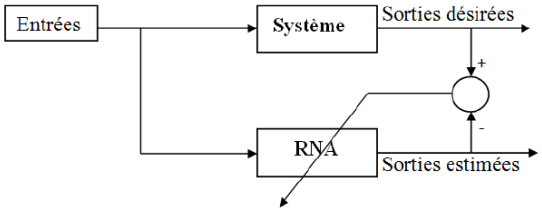

Figure 5: *Apprentissage supervisé*

2.3.3 Algorithme de rétro-propagation

Cet algorithme est utilisé pour l'apprentissage des réseaux de neurones de type FeedForward. En effet, la sortie obtenue par le réseau est comparée avec la sortie désirée, une erreur est alors obtenue. A partir de cette erreur est calculé son gradient qui est à son tour propagé de la couche de sortie vers la couche d'entrée, d'où le terme rétro-propagation. Cela permet la modification des poids synaptiques du réseau et donc l'apprentissage. L'opération est réitérée pour chaque vecteur d'entrée et cela jusqu'à ce que le critère d'arrêt soit vérifié.

Voici le déroulement de l'algorithme. On considère un réseau de neurones multicouches constitué : d'une couche d'entrée de n_0 neurones, d'une couche cachée de n_1 neurones avec une fonction d'activation sigmoïde et d'une couche de sortie de n_2 neurones avec une fonction d'activation linéaire.

2.3.3.1 Propagation

Après l'initialisation des poids synaptiques, on calcule les sorties du réseau en propageant les valeurs de l'entrée de couche en couche, la figure 6 illustre cette opération.

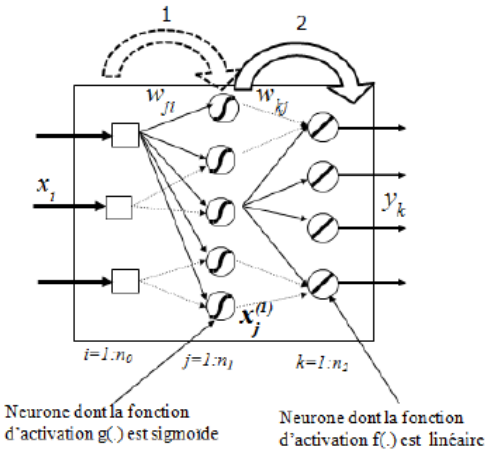

Figure 6: *Propagation des données d'entrées*

➢ **Etape 1 :**

La valeur de la sortie de chaque neurone de la couche cachée $x_j^{(1)}$ $(j = 1, \dots, n_1)$, dépend de la somme des entrées x_i $(i = 1, \dots, n_0)$ pondérées par les poids synaptiques entre la couche d'entrée et la couche cachée notés W_{ji}. Et $g()$ est la fonction d'activation des neurones de la couche cachée.

$$\begin{cases} a_j^{(1)} = \sum_{i=1}^{n_0} W_{ji} x_i \\ x_j^{(1)} = g(a_j^{(1)}) \end{cases} \quad (05)$$

➢ **Etape 2 :**

De même on calcule la valeur de la sortie de chaque neurone de la couche de sortie y_k $(k = 1, \dots, n_2)$. Elle dépend de la somme des sorties de la couche cachée qu'on a calculée dans l'étape 1 $(x_j^{(1)})$ pondérées par les poids entre la couche cachée et la couche de sortie W_{kj}.

$$\begin{cases} a_k^{(2)} = \sum_{j=1}^{n_1} W_{kj} x_j^{(1)} \\ y_k = f(a_k^{(2)}) \end{cases} \quad (06)$$

Où $f()$ étant la fonction d'activation des neurones de la couche de sortie.

➢ **Fonction coût**

On présente un exemple $x = [x_1, x_2, \dots, x_{n_0}]$ et un vecteur de sortie désiré $y^{des} = [y_1^{des}, y_2^{des}, \dots, y_{n_2}^{des}]$. On calcule le vecteur de sortie du réseau de neurones en appliquant les étapes 1 et 2. On obtient $y = [y_1, y_2, \dots, y_{n_2}]$

On calcule l'erreur entre la sortie réelle du réseau et la sortie désirée par la formule 07

$$e_k = y_k^{des} - y_k \quad (07)$$

Voici la fonction coût associée

$$J = \frac{1}{2} \sum_{k=1}^{n_2} e_k^2 \quad (08)$$

2.3.3.2 Rétro-propagation

C'est la mise à jour des poids synaptiques W_{kj} et W_{ji} du réseau en utilisant la formule 09

$$W_{nouvelle} = W_{ancienne} - \Delta W \quad (09)$$

Avec :

Si la méthode d'optimisation pour minimiser la fonction coût J est la méthode du gradient descendant alors

$$\Delta W = -\lambda \frac{\partial J}{\partial W} \quad (10)$$

Où $\frac{\partial J}{\partial W}$ est le gradient de la fonction coût par rapport aux poids synaptiques du réseau de neurones et λ est le pas du gradient.

Si la méthode d'optimisation pour minimiser la fonction coût J est la méthode de Gauss-Newton alors

$$\Delta W = -\lambda [J''(W)]^{-1} J'(W) \quad (11)$$

Où $J'(W) = \frac{\partial J}{\partial W}$ est le gradient de la fonction coût par rapport aux poids synaptiques du réseau de

neurones, $J''(W) = \left(\frac{\partial^2 J}{\partial W^2}\right)$ est la matrice Hessienne de la fonction coût par rapport aux poids synaptiques et λ est le pas du gradient.

Si la méthode d'optimisation pour minimiser la fonction coût J est la méthode de Levenberg-Marquardt alors

$$\Delta W = -[J''(W) + \lambda I]^{-1} J'(W) \qquad (12)$$

Calculons alors le gradient et la matrice Hessienne de la fonction cout par rapport aux poids synaptiques du réseau de neurones.

- **Calcul de $\frac{\partial J}{\partial W_{kj}}$ (par rapport à la deuxième couche)**

Calcul du gradient de la fonction coût par rapport aux poids synaptiques W_{kj} entre la couche cachée et la couche de sortie.

Ce gradient a pour expression :

$$\frac{\partial J}{\partial W_{kj}} = -\sum_{k=1}^{n_2} \left(y_k^{des} - y_k\right).f'\left(a_k^{(2)}\right).x_j^{(1)} \qquad (13)$$

- **Calcul de $\frac{\partial J}{\partial W_{ji}}$ (par rapport à la première couche)**

Calcul du gradient de la fonction coût par rapport aux poids de connexion W_{ji} entre la couche d'entrée et la couche.

Ce gradient a pour expression :

$$\frac{\partial J}{\partial W_{ji}} = \sum_{k=1}^{n_2} -\left(y_k^{des} - y_k\right).f'\left(a_k^{(2)}\right).W_{kj}\,.\,g'\left(a_j^{(1)}\right).x_i \qquad (14)$$

- **Calcul de $\frac{\partial^2 J}{\partial {W_{kj}}^2}$ (par rapport à la deuxième couche)**

Calcul de la matrice Hessienne de la fonction coût par rapport aux poids synaptiques W_{kj} entre la couche cachée et la couche de sortie. Elle a pour expression :

$$\frac{\partial^2 J}{\partial {W_{kj}}^2} = -\sum_{k=1}^{n_2} \left(y_k^{des} - 1\right).\left[f'\left(a_k^{(2)}\right)\right]^2.\left(x_j^{(1)}\right)^2 \qquad (15)$$

- **Calcul de $\frac{\partial^2 J}{\partial {W_{ji}}^2}$ (par rapport à la première couche)**

Calcul de la matrice Hessienne de la fonction coût par rapport aux poids de connexion W_{ji} entre la couche d'entrée et la couche cachée.

$$\frac{\partial^2 J}{\partial {W_{ji}}^2} = \sum_{k=1}^{n_2}\left(\left(\sum_{k=1}^{n_2} -\left(y_k^{des} - y_k\right).f''\left(a_k^{(2)}\right).W_{kj}\,.\,g'\left(a_j^{(1)}\right).x_i\right) W_{kj}\right) g'(a_j^{(1)})x_i \qquad (16)$$

2.4 Propriété d'approximation universelle des réseaux de neurones

Cette propriété s'énonce comme suit : « *pour toute fonction, il existe au moins un réseau de neurones non bouclé, possédant une couche de neurones cachés et un neurone de sortie linéaire, qui réalise une approximation de cette fonction et de ses dérivées successives* » [6][5]. Donc les réseaux de neurones non bouclés peuvent être tout à fait appropriés pour modéliser la fonction de prédistorsion désirée en vue de linéariser l'amplificateur. Parmi les diverses architectures de réseaux non bouclés, nous allons utiliser un réseau de neurones de type feedforward.

3. Principe de la prédistorsion à base de réseau de neurones

On peut déduire de la propriété d'« approximation universelle » des réseaux de neurones que la fonction de prédistorsion f_{PD} utilisée pour inverser la caractéristique de transfert de l'amplificateur peut être approximée à l'aide d'un réseau de neurones non bouclé de type feedforward. Pour atteindre ce but, on doit entrainer ce réseau. On va donc utiliser un algorithme d'apprentissage supervisé appelée algorithme de rétro-propagation. Voici alors les étapes à suivre pour la réalisation de la prédistorsion à base de réseau de neurones (Voir aussi la figure 7).

Etape 1 : Initialisation des poids synaptiques du réseau de type feedforward

Etape 2 : Introduction d'une base d'apprentissage. En effet, le réseau va recevoir à son entrée les données de sortie de l'amplificateur de puissance.

Etape 3 : Le réseau de neurones va estimer les données de sortie qui correspondent aux données d'entrées (les données de sortie de l'amplificateur)

Etape 4 : Comparaison entre les sorties estimées par le réseau de neurones et les sorties désirées qui sont les données d'entrée de l'amplificateur. L'erreur entre ces valeurs est définie par l'équation 17. Souvent on associe à cette erreur une fonction coût J donnée par l'équation 18.

$$e = y_{des} - y_{est} \tag{17}$$

Où

y_{des} **:** les sorties désirées

y_{est} **:** les sorties estimées par le réseau de neurones

$$J = \frac{1}{2}\sum(e)^2 \tag{18}$$

Etape 5 : La mise à jour des poids synaptiques du réseau de neurones en utilisant l'équation 19.

$$W_{nouvelle} = W_{ancienne} - \Delta W \tag{19}$$

Où

$\boldsymbol{W_{ancienne}}$ **:** ancienne valeur du poids synaptique

$\boldsymbol{W_{nouvelle}}$ **:** nouvelle valeur du poids synaptique

$\boldsymbol{\Delta W}$ **:** Valeur de correction qui dépend de l'algorithme d'optimisation utilisé pour minimiser la fonction coût J (Voir les équations 10, 11, 12).

Etape 6 : On copie le réseau entrainé dans le bloc de prédistorsion.

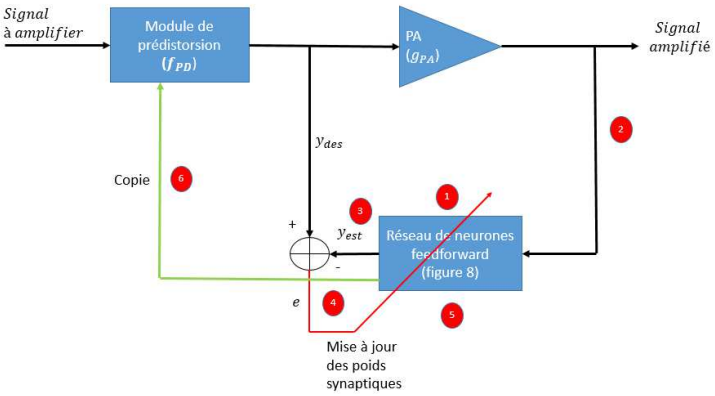

Figure 7: *Principe de la prédistorsion à base de réseau de neurones*

4. Caractéristiques du modèle neuronal proposé

Dans cet article, nous sommes dans un contexte de transmission radio-terrestre. C'est pour cette raison que le modèle d'amplificateur utilisé est le modèle de Rapp. Par conséquent, les distorsions causées par ce modèle affectent uniquement l'amplitude du signal amplifié. Donc dans notre cas, le réseau de neurones feedforward utilisé n'a qu'une seule entrée et une seule sortie. La précision de la fonction de prédistorsion obtenue dépend des caractéristiques du réseau de neurones utilisé. Dans notre cas, le réseau feedforward est composé de trois couches :

- Une couche d'entrée composée d'un seul neurone. En effet le réseau de neurones utilisé n'a qu'une seule entrée donc un neurone dans la couche d'entrée est logique.
- Une couche cachée de dix neurones ayant une fonction d'activation tangente hyperbolique. En effet, cette fonction d'activation nous permet d'initialiser aléatoirement les poids synaptiques du réseau lors de l'apprentissage, ce qui augmente la vitesse de convergence de l'algorithme d'apprentissage utilisé.
- Une couche de sortie constituée d'un seul neurone dont la fonction d'activation est linéaire.
- Dans cette étude, nous avons utilisé trois algorithmes d'apprentissage qui sont qualifiés performants :
  - Gradient descendant,
  - Gauss-Newton,
  - Levenberg-Marquardt

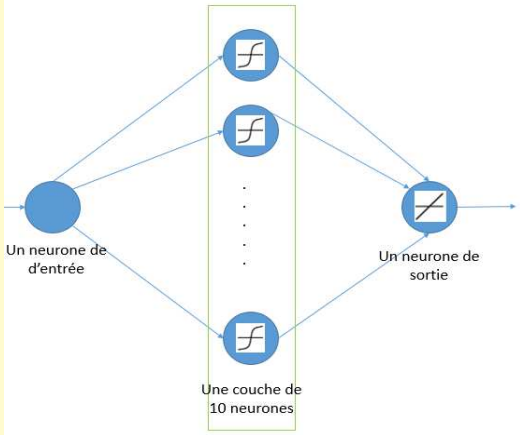

Figure 8: *L'architecture du réseau de neurones Feedforward utilisé*

On déduit de cette architecture la modélisation mathématique de la fonction de prédistorsion f_{PD}. Elle a pour expression :

$$f_{PD}(x) = \sum_{i=1}^{10}\left[w_i^{cs}\left[g(w_i^{ec}x + b_i^c)\right]\right] + b_s \quad (20)$$

Où

w_i^{cs} est le poids synaptique entre le $i^{ème}$ neurone de la couche cachée et le neurone de la couche de sortie,

w_i^{ec} est le poids synaptique entre le neurone de la couche d'entrée et le $i^{ème}$ neurone de la couche cachée,

b_i^c est le biais du $i^{ème}$ neurone de la couche cachée,

b_s est le biais du neurone de sortie,

x est le signal d'entrée du bloc de prédistorsion,

g est la fonction d'activation des neurones de la couche cachée, dans notre cas c'est une fonction tangente hyperbolique définie par $g(n) = \frac{e^n - e^{-n}}{e^n + e^{-n}}$

5. Critère de performance du modèle neuronal proposé

Pour évaluer la performance du modèle neuronal qu'on a proposé, nous avons utilisé l'Erreur Quadratique Moyenne ou MSE. Pour notre cas, elle représente la moyenne arithmétique des carrés des écarts (différences) entre la caractéristique de transfert idéal d'un amplificateur linéaire (courbe en jaune dans les figures 9, 10, 11) et la caractéristique de transfert obtenue après linéarisation d'un amplificateur non linéaire par la technique de prédistorsion à base de réseau de neurones (courbe en noire dans les figures 9, 10, 11).

6. Résultats et interprétations

Les figures 9, 10 et 11 nous montrent l'allure de la fonction de prédistorsion (courbe en rouge) obtenue à partir du modèle neuronal qu'on a proposé et la caractéristique de transfert de l'amplificateur après l'application de cette technique (courbe en noir) pour quelques algorithmes d'apprentissage. Et le tableau 1 nous donne les valeurs du MSE correspondants.

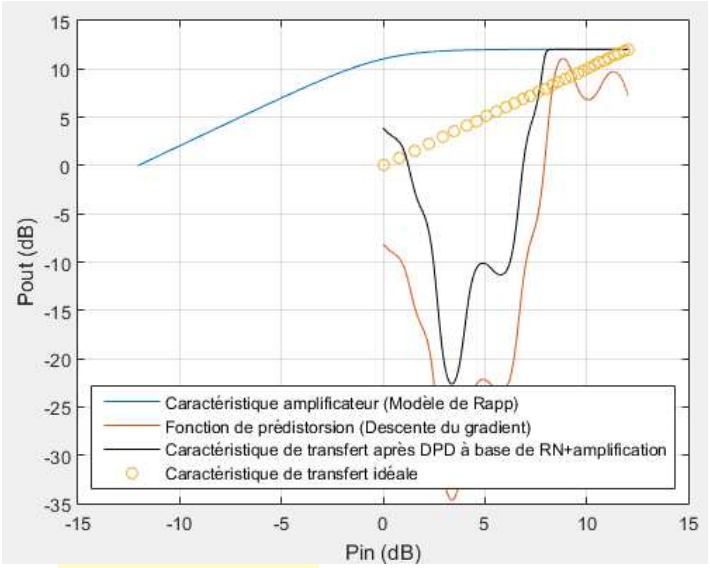

Figure 9 : *Performance du modèle neuronal après rétro-propagation associée à l'algorithme de la descente du gradient*

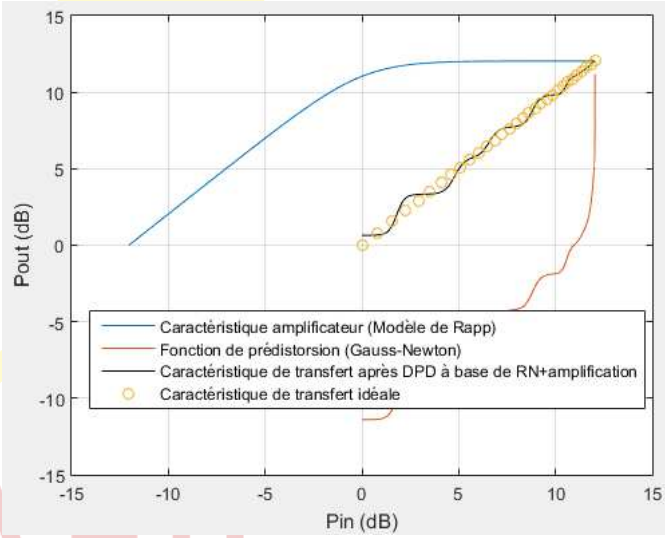

Figure 10 : *Performance du modèle neuronal après rétro-propagation associée à l'algorithme de Gauss-Newton*

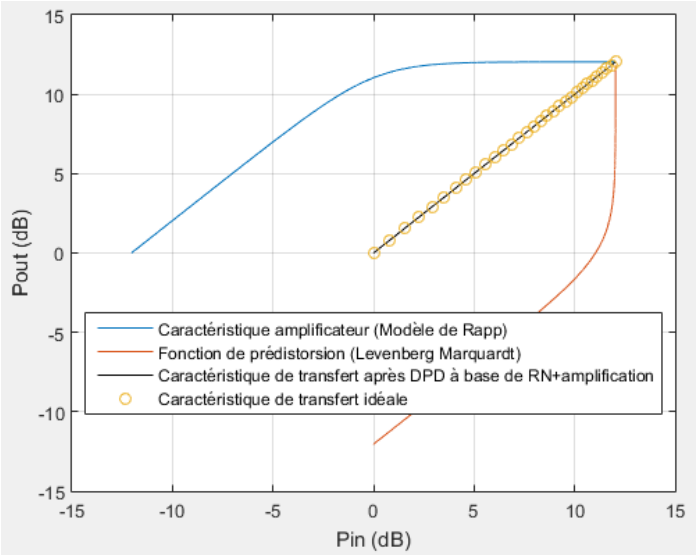

Figure 11 : *Performance du modèle neuronal après rétro-propagation associée à l'algorithme de Levenberg Marquardt*

	Descente du gradient	**Gauss-Newton**	**Levenberg Marquardt**
MSE	0.0272	1.0481×10^{-4}	4.2708×10^{-8}

Tableau 1 : *Performances obtenues pour chaque algorithme d'apprentissage*

On déduit des figures 9, 10, 11 et du tableau 1 que c'est l'algorithme de Levenberg-Marquardt qui donne la meilleur approximation de la fonction de prédistorsion. Cela vient du fait que l'algorithme de la descente du gradient converge souvent vers un minimum local mais non pas un minimum global. L'algorithme de Gauss-Newton tente de pallier cet inconvénient, cependant il se montre fragile dans sa convergence par rapport au choix des conditions initiales (initialisation des poids synaptiques). L'algorithme de Levenberg Marquardt associe les deux algorithmes précédents afin d'éviter leurs inconvénients.

7. Conclusion

Un réseau de neurones est un outil très puissant pour approximer une fonction. Dans notre cas, nous l'avons utilisé pour trouver la caractéristique de transfert inverse de l'amplificateur afin de linéariser ce dispositif à l'aide de la technique de prédistorsion. La précision de cette fonction de prédistorsion dépend des caractéristiques du réseau de neurones utilisé. Dans notre étude, nous avons proposé plusieurs modèles neuronaux basés sur divers algorithmes d'apprentissage. Et nous avons constaté que c'est l'algorithme de Levenberg Marquardt qui donne la meilleure approximation de la fonction de prédistorsion.

8. Références

[1] M. Ghannouchi, O. Hammi et M. Helaoui, *«Behavioral modeling and predistortion of wideband wireless transmitters »*, Wiley, 2015

[2] W. Chen, K. Rawat, M. Ghannouchi, *« Multiband RF Circuits and Techniques for Wireless Transmitters »,* Springer, 2016

[3] A. George, *«Intelligent Systems II : Complete Approximation by Neural Network Operators »,* Springer, 2016

[4] S. Shanmuganathan, S. Samarasinghe, *« Artificial Neural Network Modelling »*, Springer, 2016

[5] I. Rivals, L. Personnaz, G. Dreyfus, *« Modélisation, classification et commande par réseaux de neurones : principes fondamentaux, méthodologie de conception et illustrations industrielles »*, Ecole Supérieure de Physique et de Chimie Industrielles de la Ville de Paris, 2017

[6] K. Slim, *« Apprentissage multi-objectifs de réseaux de neurones pour la prévision des crues »,* Thèse de doctorat, Université Pierre et Marie Curie, Ecole des Mines de Paris , 2010