Réduction des effets des non-linéarités dans une modulation multiporteuse à l'aire de réseaux de neurones

Sylvain Tertois

N° Ordre : 2924
de la thèse

THÈSE

présentée

DEVANT L' UNIVERSITÉ DE RENNES 1

pour obtenir

le grade de : ***DOCTEUR DE L' UNIVERSITÉ DE RENNES 1***

Mention: Électronique

PAR

Sylvain TERTOIS

Équipe d'accueil : ETSN, Supélec, Campus de Rennes

École Doctorale : MATISSE

Composante Universitaire : S.P.M.

RÉDUCTION DES EFFETS DES NON-LINÉARITÉS DANS UNE MODULATION MULTIPORTEUSE A L'AIDE DE RÉSEAUX DE NEURONES

SOUTENUE LE 12 décembre 2003 devant la commission d'Examen

COMPOSITION DU JURY:

Directeur de thèse:	Olivier BONNAUD	Professeur à l'Université de Rennes 1
Rapporteurs:	Gilles BUREL	Professeur à l'Université de Bretagne Occidentale
	Daniel ROVIRAS	Professeur à l'INPT (Toulouse)
Examinateurs:	Jean-François HELARD	Professeur à l'INSA (Rennes)
	Yves LOUET	Enseignant-chercheur à Supélec (Rennes)
	Annick LE GLAUNEC	Enseignant-chercheur à Supélec (Rennes)
Membre Invité:	Gilles VAUCHER	Professeur à Supélec (Rennes)

Table des matières

Liste des illustrations

Liste des tableaux

Liste des équations

Remerciements

Je tiens tout d'abord à remercier messieurs Yves Quenec'hdu et Bernard Loriferne pour leur accueil sur le campus de Rennes de Supélec.

Je remercie Olivier Bonnaud, professeur à l'Université de Rennes 1, pour avoir accepté de diriger ces travaux.

Je remercie Gilles Burel, professeur à l'Université de Bretagne Occidentale, et Daniel Roviras, professeur à l'Institut National Polytechnique de Toulouse pour avoir accepté d'être rapporteurs pour cette thèse, ainsi que Jean-François Hélard, professeur à l'Institut National des Sciences Appliquées de Rennes, qui a accepté d'être examinateur.

Je remercie vivement Annick Le Glaunec, Yves Louët et Gilles Vaucher pour leur encadrement des travaux, leurs suggestions et leur aide, ainsi que leur conseils lors de la rédaction du manuscrit.

Je remercie tous les membres de l'équipe ETSN de Supélec, ainsi que les thésards de Supélec, campus de Rennes, pour leur sympathie et pour avoir rendu aussi agréable l'environnement de travail. Je voudrais plus particulièrement remercier Pierre Leray pour ses déplantages des outils sur les stations Sun et Nicolas, Cécile, Thomas et Olivier pour leur contribution à la bonne ambiance parmi les thésards.

Je voudrais enfin remercier les services technique et logistique de Supélec, campus de Rennes, sans qui les conditions de travail ne seraient pas aussi bonnes.

Résumé

Ce mémoire présente les travaux effectués au sein de l'équipe ETSN de Supélec, campus de Rennes, sur la réduction des effets des non linéarités dans une modulation OFDM, à l'aide de réseaux de neurones.

Tout d'abord, le mémoire commence par une introduction aux communications numériques et en particulier à la modulation OFDM. Aujourd'hui, plusieurs standards reposent sur cette technique de transmission, en particulier en raison de la simplicité de l'égalisation du canal, et donc la possibilité de transmettre avec plus d'efficacité des données sur des canaux multitrajets. Cependant le signal OFDM temporel est particulièrement sensible aux non-linéarités présentes dans l'amplificateur d'émission et diverses techniques sont étudiées pour diminuer ces effets.

Ensuite, les réseaux de neurones sont présentés, ainsi que leur utilisation dans le domaine de l'approximation de fonctions. Après avoir décrit les deux modèles de réseaux de neurones les plus courants, les réseaux d'ordre supérieur, tels que le RPN, sont introduits. Les techniques d'apprentissage de ces différentes architectures de réseaux de neurones sont également décrites.

Dans les différents correcteurs étudiés dans cette thèse, le réseau de neurones est placé dans le récepteur, après l'égalisation de canal. Son objectif est de corriger le signal reçu afin de compenser les effets des non-linéarités. Dans un premier temps le réseau de neurones est placé dans le domaine fréquentiel. Dans un système OFDM à 4 porteuses avec une modulation MAQ16, un amplificateur de type SSPA, un recul de 0 dB et pour un taux d'erreur binaire de 10^{-2}, le correcteur avec un réseau RPN apporte un gain de 1,5 dB de rapport signal sur bruit. Cependant des difficultés apparaissent durant la phase d'apprentissage du réseau de neurones avec un nombre de porteuses supérieur.

Pour palier ce défaut, les réseaux de neurones décrits précédemment sont simplifiés en étant placés dans le domaine temporel. Ce système est plus proche des solutions déjà proposées pour la compensation des non-linéarités dans une modulation monoporteuse, avec toutefois des différences au niveau de l'égalisation du canal et de la nature de la fonction que doit accomplir le réseau de neurones. Un correcteur basé sur un réseau RPN a montré de très bonnes performances, même en augmentant le nombre de porteuses. Un gain de 8 dB a été mesuré pour un taux d'erreur binaire de 10^{-2} dans un système OFDM à 48 porteuses, une modulation MAQ16 et un amplificateur de type SSPA avec un recul de 0 dB. Le système présenté permet donc dans ces conditions de diviser la puissance de l'amplificateur, et donc sa consommation d'énergie, par un facteur supérieur à 4 tout en conservant la même qualité de transmission.

Le correcteur à RPN dans le domaine temporel est ensuite simulé sur un canal multitrajet, afin de vérifier que la compensation reste efficace dans le cas d'un canal sévère. Enfin les deux approches proposées (fréquentielle et temporelle) sont comparées, au niveau des performances obtenues et de la puissance de calcul nécessaire dans le récepteur. Une comparaison avec une autre approche proposée dans la littérature est également présentée. Le correcteur temporel basé sur un RPN est bien moins complexe que le système cité, au détriment d'une légère dégradation des performances.

Ce mémoire se conclut par quelques perspectives de recherche pouvant prolonger les travaux accomplis durant cette thèse.

Introduction Générale

Les télécommunications font partie des technologies qui ont révolutionné notre mode de vie au vingtième siècle. Du télégraphe à l'Internet, de la TSF au téléphone cellulaire, les progrès établis en la matière sont spectaculaires. Les informations transmises étaient tout d'abord codées en morse, puis des techniques de modulation et de codage analogiques ont permis de transmettre du son, puis des images. Ensuite la venue des techniques numériques a considérablement augmenté le débit et la qualité des informations à transmettre d'un point à un autre.

Parallèlement, le développement rapide de la microélectronique et des capacités de miniaturisation permettent aujourd'hui la mise en oeuvre de techniques complexes dans des appareils de taille réduite. Cependant l'augmentation des besoins en débit se heurte à la nature des canaux eux-mêmes. En effet, dans des applications telles que la télédiffusion à grande échelle ou un réseau informatique radio à l'intérieur d'un bâtiment, le canal est de type multitrajet. Le signal est réfléchi en plusieurs endroits, et des échos apparaissent et créent des perturbations dont l'influence augmente avec le débit de transmission. Parmi les solutions étudiées pour palier ce problème, les modulations multiporteuses sont étudiées depuis les années soixante et sont particulièrement adaptées à ces canaux sévères. Un des obstacles principaux à leur mise en oeuvre était la complexité des appareils d'émission et de réception, mais les progrès accomplis depuis en électronique ont rendu leur réalisation possible. Aujourd'hui plusieurs normes telles que DAB (Digital Audio Broadcasting, norme de diffusion audio numérique), DVB-T (Digital Video Broadcasting - Terrestrial, norme de télévision numérique sur canal hertzien) et ADSL (Asymetric Digital Subscriber Line) reposent sur les modulations multiporteuses.

Toutefois, un des inconvénients majeurs des modulations multiporteuses réside dans le fort facteur de crête du signal temporel. Défini comme le rapport entre la puissance maximale et la puissance moyenne de ce signal, il en caractérise les fluctuations, ces dernières pouvant être relativement importantes. Cet aspect devient néfaste dès lors qu'il s'agit d'amplifier le signal temporel. En effet les caractéristiques des amplificateurs de puissance ne sont pas parfaitement linéaires et le signal ainsi amplifié présente des distorsions, véhiculant ainsi des erreurs de transmission. Actuellement de nombreux travaux de recherche visent à réduire ces erreurs, soit en diminuant le facteur de crête du signal multiporteuse, soit en compensant les erreurs introduites.

Ce thème a déjà fait l'objet de travaux dans l'équipe ETSN (Électronique, Traitement du Signal et Neuromimétisme) de Supélec, campus de Rennes. Dans sa thèse présentée en octobre 2000, Yves Louët s'est intéressé à une technique de codage canal particulière côté émission qui permet une réduction du facteur de crête du signal temporel [LOUE00]. L'objet de la thèse présentée dans ce mémoire est l'utilisation d'un réseau de neurones côté récepteur dont l'objectif est de réduire les effets des non-linéarités de l'amplificateur sur la transmission multiporteuse.

Les réseaux de neurones font également partie des thèmes de recherche de l'équipe ETSN, et constituent un ensemble de techniques de traitement du signal qui s'inspirent du vivant, et en particulier des systèmes nerveux. Ces techniques ont profité elles aussi du développement de la micro-électronique et constituent des systèmes capables d'accomplir des tâches complexes

telles que la reconnaissance de visage ou l'écriture manuscrite. Elles trouvent de plus une place dans le domaine des télécommunications, en particulier dans les systèmes comportant des non-linéarités. Les travaux présentés dans ce mémoire ont donc pour but d'étudier leur emploi dans le contexte des modulations multiporteuses.

Ce mémoire est organisé en 5 chapitres.

Dans le premier chapitre on présente tout d'abord les communications numériques et les notions de télécommunications qui serviront au cours de cette thèse. Les canaux multitrajets et la modulation OFDM (Orthogonal Frequency Division Multiplexing), qui est la modulation multiporteuse sur laquelle portent ces travaux, seront également introduits. Ensuite le problème du facteur de crête dans une modulation OFDM est évoqué, ainsi que ses conséquences dans une chaîne de transmission présentant des non-linéarités. Enfin un aperçu des méthodes déjà existantes pour tenter de résoudre le problème est présenté. La plupart des techniques présentées s'appuient sur le codage canal ou sur une modification de l'émetteur. Quelques autres techniques reposent elles sur une modification du récepteur et permettent ainsi leur utilisation dans des systèmes déjà normalisés puisqu'aucune modification du codage ou de l'émetteur n'est nécessaire.

Le chapitre 2 est une introduction aux réseaux de neurones, et en particulier aux perceptrons multicouches et aux GRBF qui sont les modèles les plus couramment employés. Ce chapitre insiste sur la mise en oeuvre de ces méthodes neuronales, tout en donnant des références à des études plus théoriques des outils présentés. Les architectures des réseaux sont présentées, ainsi que leurs méthodes d'apprentissages. Les algorithmes d'optimisation servant à l'apprentissage ne sont pas abordés dans ce chapitre, mais dans l'annexe A. Les réseaux d'ordre supérieur sont ensuite décrits, car ils sont au centre des travaux menés dans le cadre de cette thèse. En réalisant des produits entre leurs entrées, ces réseaux peuvent utiliser des corrélations d'ordre supérieur entre les entrées, et sont ainsi plus adaptés à certains problèmes rencontrés en traitement du signal. Différentes architectures d'ordre supérieur sont présentées, avec leur méthode d'apprentissage.

Le chapitre 3 présente la méthodologie qui a été employée pour utiliser les outils neuronaux dans un système de transmission multiporteuse. La caractéristique principale est que le réseau de neurones est placé dans le récepteur, après l'égalisation du canal, et dans le domaine fréquentiel. Une étude théorique de l'influence des non-linéarités dans le domaine fréquentiel permet d'établir des règles de symétrie, ainsi qu'une expression de la forme de la fonction que devra réaliser le réseau de neurones. Ensuite l'apprentissage et l'exploitation de différents correcteurs basés sur des réseaux de neurones sont présentés. Les perceptrons multicouches n'ont pas permis d'obtenir des résultats satisfaisants, mais par contre un réseau d'ordre supérieur a montré de bonnes performances dans un système à 4 porteuses. Des études ont été réalisées avec différents types de non-linéarités, et on montre qu'il existe un rapport signal sur bruit optimal pour l'apprentissage du réseau de neurones. Le nombre de porteuses du système OFDM simulé est faible par rapport aux applications pratiques de cette modulation multiporteuses, et lorsqu'on l'augmente, l'apprentissage du réseau de neurones devient difficile et le gain qu'il apporte diminue.

Ainsi dans le chapitre 4, le correcteur est légèrement modifié afin de faire fonctionner le réseau de neurones dans le domaine temporel. Tout d'abord un correcteur idéal basé sur le modèle de l'amplificateur est simulé, afin de prouver la faisabilité d'une telle approche. On parvient à obtenir des performances satisfaisantes à condition d'ajouter une marge de saturation au

modèle selon un procédé qui sera décrit dans ce chapitre. Ensuite des correcteurs basés sur des réseaux de neurones sont entraînés et simulés. Ceux qui sont basés sur des perceptrons multicouches aussi bien que des réseaux d'ordre supérieur ont montré de bonnes performances, principalement pour ces derniers. Les réseaux utilisés sont plus simples que ceux qui étaient employés dans le domaine fréquentiel, et leur complexité est indépendante du nombre de porteuses. De plus leur apprentissage est bien plus rapide que celui des réseaux de neurones présentés au chapitre 3. Des simulations montrent qu'en augmentant ce nombre de porteuses, les performances du correcteur restent bonnes, et son utilisation peut être envisagée dans des applications OFDM existantes, comme par exemple la norme de réseau informatique sans fil HiperLAN/2.

Dans le chapitre 5, nous apportons des idées de solutions aux problèmes que pourraient poser la mise en oeuvre des correcteurs présentés dans les deux chapitres précédents. Leurs performances sur un canal multitrajet sont étudiés, afin de s'assurer qu'ils préservent les avantages de la modulation multiporteuse sur ces canaux sévères, et différents moyens de réaliser leur apprentissage sont proposés. Ensuite ces deux correcteurs sont comparés selon plusieurs critères, tels que leurs performances en compensation des erreurs dues aux non-linéarités, et également leur puissance de calcul requise, ainsi que l'évolution de cette dernière en augmentant le nombre de porteuses. Enfin, une comparaison est faite avec un système présenté dans la littérature.

Ce mémoire se termine par une conclusion et des perspectives de recherches qu'il est possible de mener afin de prolonger le travail effectué durant cette thèse.

Afin de faciliter la lecture de ce mémoire, une présentation des conventions utilisées dans les équations se situe après cette introduction. De plus, à la fin de ce mémoire se trouvent un glossaire qui regroupe tous les sigles présents dans ce document, une liste des notations employées dans les équations et un index.

Notations

Dans un souci de cohérence du mémoire, une convention de notation commune à toutes les parties a été adoptée. En effet les habitudes ne sont pas les mêmes dans les différents domaines abordés (télécommunications, réseaux de neurones). Le choix suivant a donc été fait :

Les valeurs et fonctions scalaires sont en italiques. Une fonction temporelle est en minuscule (signal temporel $a(t)$) avec éventuellement un indice ($a_j(t)$). Les fonctions fréquentielles sont en majuscule : la transformée de Fourier de $a_j(t)$ se note $A_j(f)$. Les constantes caractéristiques d'un système sont sous la forme d'une majuscule avec une minuscule en indice : le nombre de porteuses N_p, la durée d'un symbole T_s. Les signaux discrets et suites de valeurs scalaires sont sous la forme d'une minuscule avec une ou plusieurs minuscules en indice : les symboles transmis c_k. Un indice servant de compteur est l'une des lettres i, j, k ou l. Les lettres n et m peuvent représenter une limite pour l'évolution de ces compteurs (par exemple $k = 0 \dots n$).

Les vecteurs sont en gras non italiques et en minuscules, généralement avec un ou plusieurs indices : le symbole OFDM $\mathbf{c_j}$. Les composantes de ces vecteurs sont décris avec la même lettre en italique, avec un indice supplémentaire : les composantes de $\mathbf{c_j}$ sont $c_{j,k}$. Les fonctions retournant un scalaire prenant un vecteur comme paramètre sont en italiques avec le paramètre en gras ($y_k = f(\mathbf{x})$) tandis que les fonctions vectorielles sont en gras ($\mathbf{y} = \mathbf{f}(\mathbf{x})$).

Les matrices sont en gras non italiques et en majuscules, éventuellement avec un ou plusieurs indices : la matrice de poids $\mathbf{W_{ij}}$. Les éléments d'une matrice sont décris avec la même lettre en minuscule et deux indices supplémentaires : les composants de $\mathbf{W_{ij}}$ sont $w_{ij,k,l}$.

Dans un souci de clarté, si trop d'indices sont nécessaires, certains peuvent être mis en exposant entre parenthèses. Par exemple la matrice de poids peut être notée $\mathbf{W^{(ij)}}$, et donc ses composantes $w_{k,l}^{(ij)}$.

Un index des notations est présent à la fin de ce document, et permet au lecteur de retrouver la définition d'une notation particulière présente dans une équation, et la page à laquelle elle est définie.

La valeur absolue du réel x est notée $|x|$, de même que le module du complexe y, noté $|y|$. Le conjugué du complexe y est noté $\overline{y}$, sa partie réelle $\Re(y)$ et sa partie imaginaire $\Im(y)$. La norme euclidienne du vecteur réel ou complexe $\mathbf{x}$ est notée $\|\mathbf{x}\|$, et la transposée d'une matrice $\mathbf{A}$ est notée $\mathbf{A}^T$. Enfin la convolution des deux signaux x et y est notée $x \star y$.

Chapitre 1. Modulations multiporteuses et non linéarités

Introduction

Ce chapitre a pour but d'introduire les modulations multiporteuses, et en particulier les notions qui serviront à la bonne compréhension de cette thèse. Après un aperçu sur les modulations numériques en général, les modulations multiporteuses sont présentées, ainsi que leurs avantages sur les canaux sélectifs en fréquence. Cependant elles sont particulièrement sensibles aux non-linéarités inhérentes aux composants utilisés en pratique, et de nombreuses méthodes sont développées pour limiter les erreurs qu'elles engendrent dans une transmission, comme on le verra en dernière partie de ce chapitre.

Les sections 1 et 2 étant respectivement des introductions aux communications numériques et aux modulations multiporteuses, elles peuvent être sautées en fonction du niveau de compétences du lecteur sur ces sujets.

1. Transmission numérique

Le but d'une modulation est de transmettre des informations d'un émetteur à un récepteur, à travers un canal de transmission. Ce canal possède un certain nombre de caractéristiques et de contraintes qu'il faut prendre en compte. Par exemple un canal sélectif en fréquence atténue le signal dans certaines bandes de fréquences et l'amplifie dans d'autres bandes, et ces perturbations doivent être prises en compte par le système de transmission. Ainsi les données numériques doivent subir un certain nombre de transformations avant d'être transmises, et une autre série de transformations est effectuée dans le récepteur pour obtenir à nouveau les données numériques envoyées. Les différentes étapes seront explicitées successivement dans ce chapitre. Le schéma ci-dessous résume l'ensemble de ces étapes.

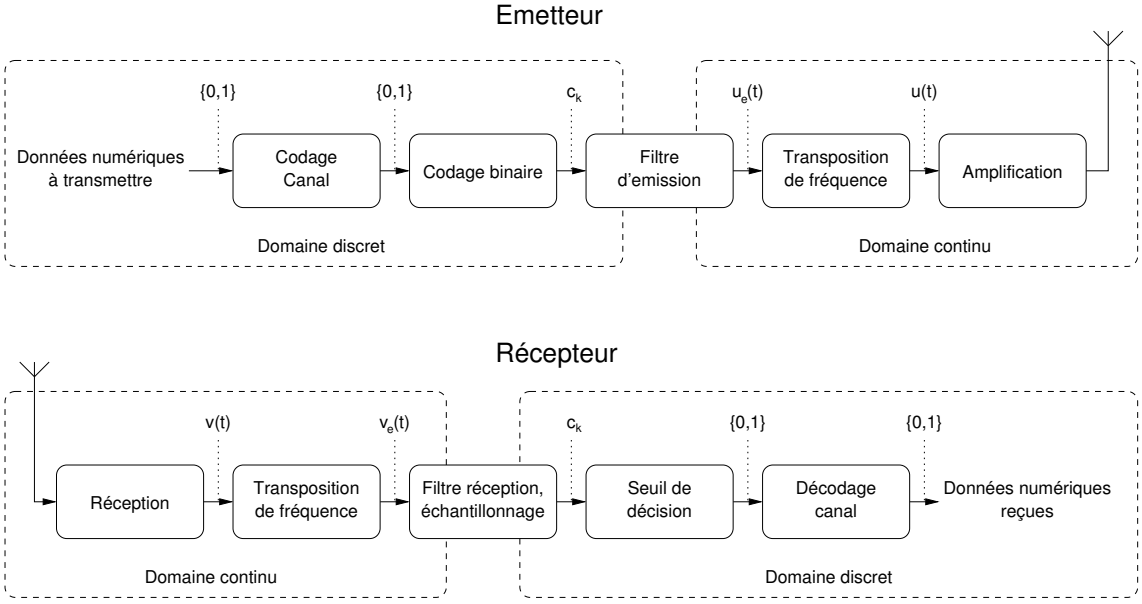

Figure 1.1. Étapes d'une chaîne de transmission numérique

1.1. Codage canal

L'ensemble des données numériques à transmettre est codé en une suite de 0 et de 1 par un processus qui dépend de la nature des données, et qui généralement supprime de la redondance éventuellement présente dans celles-ci. Par exemple pour la transmission d'images on peut utiliser l'algorithme de compression avec pertes JPEG. Le signal à transmettre peut être représenté par une source binaire, dont la sortie est un signal discret constitué des deux éléments binaires 0 et 1 (bits). Cette source possède un débit, qui s'appelle débit d'information binaire de la chaîne de transmission, et qui est égal au nombre d'éléments binaires transmis par la source par unité de temps.

En pratique des erreurs peuvent se produire durant la communication, et elles sont principalement dues au bruit et aux interférences produites par le canal de transmission lui-même. Pour y remédier, on utilise un codage correcteur d'erreurs : des bits de redondance sont ajoutés aux informations numériques à transmettre, et ceux-ci permettent au récepteur de détecter et/ou corriger des erreurs. Ces codes ne sont pas abordés dans ce document, mais le lecteur intéressé peut se référer à [COHE92].

1.2. Codage binaire à symbole

Le codage binaire à symbole [1] est l'étape qui génère un signal discret à partir des données numériques. Chaque élément c_k de ce signal est appelé symbole , peut être réel ou complexe, et est associé à un ou plusieurs bits issus de la source d'informations. On définit alors un second débit sur le canal, le débit symbole, qui est le nombre de symboles transmis par unité de temps. Il est mesuré en bauds et est égal au débit binaire divisé par le nombre de bits représenté par chaque symbole.

Le système le plus simple que l'on puisse envisager est la modulation d'amplitude à deux états. Chaque symbole c_k du signal discret correspond à un bit de donnée numérique à transmettre. Si on note $A \in \mathbb{R}$ l'amplitude du signal, les deux valeurs possibles pour c_k sont :

$$c_k \in \{-A, A\} \tag{1.1}$$

Ces deux valeurs correspondent respectivement à 0 et 1. Il est également possible de coder plus de bits dans un même symbole, en définissant plus de valeurs possibles. Par exemple pour coder deux bits dans un seul symbole, on peut utiliser une modulation d'amplitude à 4 états :

$$c_k \in \{-3A, -A, A, 3A\} \tag{1.2}$$

Par extension on peut construire une modulation d'amplitude à 2^n états, où chaque symbole code donc n bits.

Les symboles c_k peuvent être complexes, et il est donc envisageable de coder l'information dans la phase des symboles. Certaines applications nécessitent en effet un signal avec un module constant. Si on appelle m le nombre d'états, l'ensemble des symboles possibles est :

[1] cette étape sera appelée "codage binaire" dans le reste de ce mémoire, afin de ne pas confondre avec le "codage canal", aussi appelé "codage correcteur d'erreur" évoqué dans le paragraphe précédent.

$$c_k \in \{Ae^{\frac{2ik\pi}{m}+\varphi_0}, k = 0 \dots m-1\} \tag{1.3}$$

où φ_0 est la phase du premier symbole et m est une valeur de la forme 2^n. Ainsi chaque symbole c_k code n bits et on note MDPm une modulation de phase à m états.

Enfin il est possible de coder de l'information à la fois dans les parties réelle et imaginaire du symbole c_k. Cette technique est appelée Modulation d'Amplitude en Quadrature , et est notée MAQm, où m est le nombre d'états de la modulation. Les MAQ les plus courantes utilisent le même codage sur les deux parties réelle et imaginaire. Dans ce cas m est de la forme 2^{2n}, et chaque symbole code $2n$ bits : n bits dans la partie réelle, et n dans la partie imaginaire. Par exemple la modulation MAQ16 utilise deux modulations d'amplitude à 4 états :

$$c_k = a_k + ib_k, a_k, b_k \in \{-3A, -A, A, 3A\} \tag{1.4}$$

Un codage binaire peut se représenter de manière graphique, appelée constellation, dont chaque point correspond à un symbole c_k, à côté duquel on indique éventuellement la donnée numérique que le symbole code. Par exemple les constellations des codages MDP8 et MAQ16 peuvent être représentés de la forme suivante, dans le cadre d'un codage de Gray :

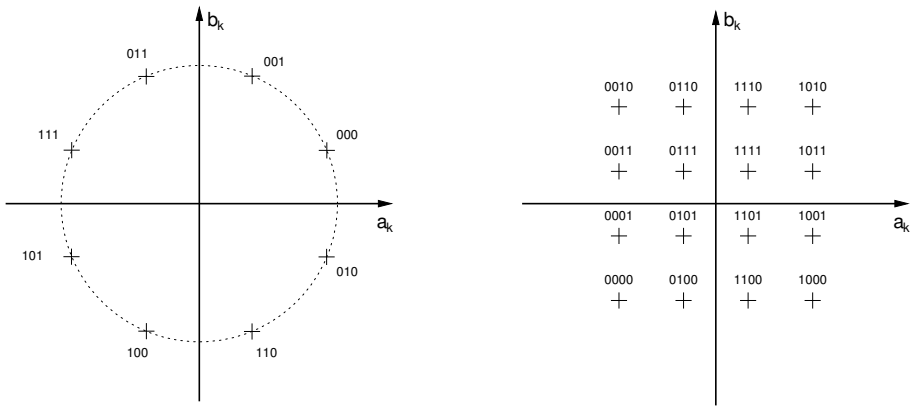

Figure 1.2. Exemples de constellations

1.3. Forme d'onde et filtre d'émission

Le canal de transmission étant un milieu continu, avant de pouvoir y transmettre les symboles c_k il faut obtenir un signal continu par interpolation. Les symboles sont cadencés par une horloge à la fréquence $F_S = 1/T_S$, où T_S est la durée d'un symbole, et une forme d'onde $h(t)$ permet d'interpoler le signal discret. $h(t)$ est une fonction non nulle sur $[0, T_S]$ et comme son nom l'indique donne la forme au signal continu :

$$u_e(t) = \sum_{k=0}^{\infty} c_k h(t - kT_S) \tag{1.5}$$

Une forme d'onde classique est tout simplement le rectangle de durée T_S :

$$h(t) = \begin{cases} 1 & 0 \leq t < T_S \\ 0 & \text{ailleurs} \end{cases} \tag{1.6}$$

Le signal ainsi généré a un spectre infini. En pratique il est impossible d'utiliser tout le spectre dans un canal réel, et il est alors nécessaire d'ajouter après cette forme d'onde un filtre basse fréquence, qui servira à limiter la bande passante du signal émis. Ce filtre est appelé filtre d'émission, et on notera $g_e(t)$ sa réponse impulsionnelle. Il est placé après la forme d'onde, et souvent dans un souci de simplification on appelle "filtre d'émission" l'ensemble forme d'onde - filtre d'émission, qui a donc pour réponse impulsionnelle $(h \star g_e)(t)$.

1.4. Transposition de fréquence et amplification

La transposition de fréquence est nécessaire dans le cas d'une transmission radio. En effet un canal radio est caractérisé par une bande de fréquences précise, et afin de ne pas perturber les communications sur les autres canaux radio, il faut s'assurer que la transmission n'utilise que cette bande de fréquence. La largeur de cette bande Δf est souvent faible devant sa fréquence centrale f_0, et ainsi le signal qui y est propagé est dit à bande étroite. Le signal provenant du filtre d'émission est quand à lui un signal basse fréquence, dit signal en bande de base. La modulation, ou transposition de fréquence, consiste donc à décaler la fréquence centrale du signal pour respecter les caractéristiques imposées par le canal. La figure 1.3 montre la forme des densités spectrales de puissance (DSP) du signal avant et après transposition de fréquence.

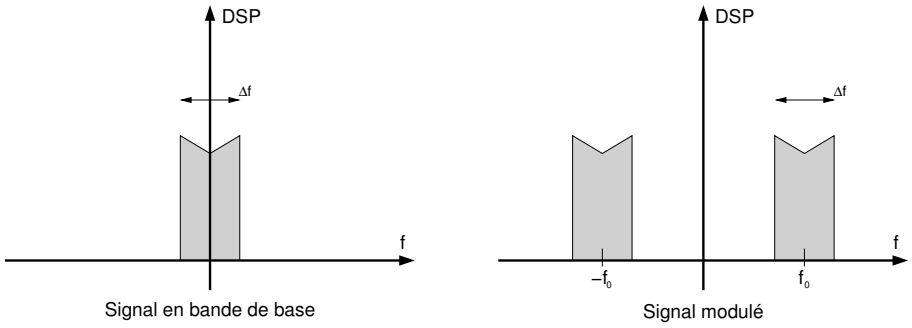

Figure 1.3. Spectres de signaux avant et après transposition de fréquence

Parmi les opérations classiques de transposition à la fréquence f_0, la modulation d'amplitude est la plus simple. Pour cela le signal à moduler est multiplié par un signal sinusoïdal appelé

porteuse. Le signal obtenu est le signal d'origine dont le spectre est décalé autour de la fréquence de la porteuse, et a pour largeur de bande celle du signal en bande de base. Si on appelle $a(t)$ le signal à moduler et f_0 la fréquence de la porteuse, le signal modulé $u(t)$ est :

$$u(t) = a(t)\sin(2\pi f_0 t) \tag{1.7}$$

Il est possible de créer deux porteuses orthogonales à la même fréquence en les déphasant de $\frac{\pi}{2}$. Cette méthode s'appelle modulation en quadrature, et permet alors de moduler deux signaux $u_c(t)$ et $u_s(t)$ avec chacune de ces porteuses et ainsi doubler la quantité d'information transmise dans la même bande de fréquence. Comme on va le voir dans la section suivante, l'orthogonalité des porteuses assure que les deux signaux seront séparables à la réception :

$$u(t) = u_c(t)\cos(2\pi f_0 t) - u_s(t)\sin(2\pi f_0 t) \tag{1.8}$$

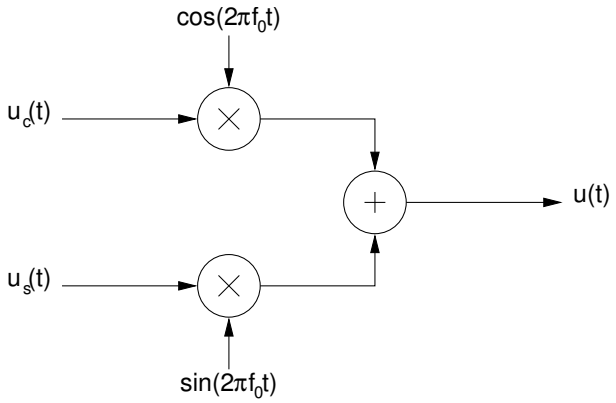

Figure 1.4. Schéma de réalisation d'un modulateur en quadrature

Pour simplifier les notations, les deux signaux $u_c(t)$ et $u_s(t)$ peuvent être regroupés en un seul, appelé enveloppe complexe :

$$u_e(t) = u_c(t) + iu_s(t) \tag{1.9}$$

Dans ce cas la modulation peut s'écrire :

$$u(t) = \Re(u_e(t)e^{2i\pi f_0 t}) \tag{1.10}$$

Si l'on regroupe le filtre d'émission et la transposition de fréquence, on peut exprimer le signal modulé sous la forme :

$$u(t) = \Re(e^{2i\pi f_0 t}\sum_{k=0}^{\infty} c_k(h \star g_e)(t - kT_S)) \tag{1.11}$$

Une autre notation peut être utilisée pour exprimer le signal modulé, à l'aide d'une base de signaux élémentaires :

$$\psi_k(t) = e^{2i\pi f_0 t}(h \star g_e)(t - kT_S) \tag{1.12}$$

En utilisant cette base, l'expression du signal modulé devient :

$$u(t) = \Re(\sum_{k=0}^{\infty} c_k \psi_k(t)) \tag{1.13}$$

Cette notation servira lors de la présentation des modulations multiporteuses.

Un amplificateur est enfin nécessaire pour augmenter la puissance du signal afin que son niveau soit suffisant au niveau du récepteur[2] , compensant ainsi les pertes en espace libre. Dans le cas d'un canal radio l'amplificateur est relié à une antenne qui rayonne et crée ainsi le signal radio. Dans le cas d'un canal filaire l'amplificateur est relié au câble.

1.5. Canal, réception et démodulation

Le canal de propagation perturbe le signal, en le déformant et en y ajoutant du . Ces deux aspects seront abordés plus loin dans ce mémoire, nous supposerons dans un premier temps que le canal est parfait. Le récepteur recueille le signal transmis, par l'intermédiaire d'une antenne pour un canal radio ou directement depuis le câble pour une transmission filaire. Une fois le signal ré-amplifié il est nécessaire de le démoduler, c'est-à-dire de faire une nouvelle transposition de fréquence afin d'obtenir un signal en bande de base.

Si l'on connaît la fréquence de la porteuse f_0, une démodulation que l'on appelle cohérente permet de retrouver le signal d'origine. Pour cela le signal reçu $u(t)$ est à nouveau multiplié par une sinusoïde à la fréquence porteuse f_0 et le signal obtenu $s(t)$ est alors la somme de deux signaux : le signal en bande de base qui contient l'information, et un second signal modulé à la fréquence $2f_0$. En réalisant un filtrage passe-bas le signal en bande de base $A(t)$ peut être isolé :

$$s(t) = u(t)\sin(2\pi f_0 t) = A(t)\sin^2(2\pi f_0 t) = \frac{A(t)}{2} - A(t)\frac{\cos(4\pi f_0 t)}{2} \tag{1.14}$$

Cette démodulation s'applique dans le cas d'une modulation classique, c'est à dire avec une seule porteuse. Dans le cas d'une modulation en quadrature, les deux signaux $u_c(t)$ et $u_s(t)$ peuvent être retrouvés au niveau du récepteur en réalisant deux démodulations, avec deux porteuses déphasées également de $\frac{\pi}{2}$:

$$\begin{aligned}
s_c(t) &= u(t)\cos(2\pi f_0 t) = u_c(t)\cos^2(2\pi f_0 t) - u_s(t)\sin(2\pi f_0 t)\cos(2\pi f_0 t) \\
&= \frac{u_c(t)}{2} + u_c(t)\frac{\cos(4\pi f_0 t)}{2} - u_s(t)\frac{\sin(4\pi f_0 t)}{2} \\
s_s(t) &= -u(t)\sin(2\pi f_0 t) = -u_c(t)\cos(2\pi f_0 t)\sin(2\pi f_0 t) + u_s(t)\sin^2(2\pi f_0 t) \\
&= \frac{u_s(t)}{2} - u_s(t)\frac{\cos(4\pi f_0 t)}{2} - u_c(t)\frac{\sin(4\pi f_0 t)}{2}
\end{aligned} \tag{1.15}$$

Au moyen d'un filtre passe-bas les deux signaux en bande de base sont isolés.

[2]Dans le cas du canal hertzien, l'atténuation en espace libre du signal radio peut être très importante, et dépend de la fréquence de modulation. L'effet est moindre dans un canal filaire mais tout de même sensible sur de longues distances

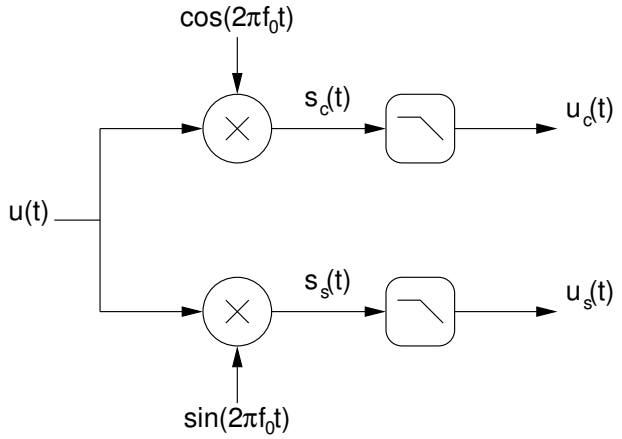

Figure 1.5. Schéma de réalisation d'un démodulateur en quadrature

1.6. Filtre de réception

Le signal démodulé est un signal continu, mais le récepteur va devoir réaliser un échantillonnage afin de déterminer les d'éléments binaires transmis. Cependant avant l'échantillonnage, on montre [GLAV96] qu'il faut réaliser un filtrage adapté à l'émetteur pour une réception optimale des symboles transmis. Dans le cas où aucun filtre d'émission n'est employé (c'est à dire qu'on utilise simplement une forme d'onde, et que $g_e(t)$ est un Dirac), le filtre de réception adapté à la forme d'onde $h(t)$ a pour réponse impulsionnelle :

$$g_r(t) = h(t_0 - t) \tag{1.16}$$

Où t_0 est l'instant d'échantillonnage. Le système de réception est très simple dans ce cas, car le signal reçu à un instant donné après ce filtrage correspond directement à un unique symbole, et celui-ci peut alors être décodé. Par contre lorsqu'un filtre d'émission est utilisé la réponse de ce filtre est généralement plus longue que T_S, et le signal reçu à un instant t ne dépend plus d'un seul symbole émis, mais également des autres symboles. Ce phénomène est appelé interférence entre symboles (IES). Pour annuler cette interférence, il faut qu'à l'instant d'échantillonnage on ne prélève que le symbole émis, et annuler l'influence due aux autres symboles. C'est-à-dire qu'il faut que la réponse impulsionnelle $r(t)$ de la chaîne de transmission complète ($r = h \star g_e \star g_r$) vérifie :

$$r(t_0 + kT_S) = p_0\delta_{0k} \; \forall k \tag{1.17}$$

où p_0 est un réel, t_0 et T_S sont respectivement l'instant et la période d'échantillonnage et δ_{0k} est le symbole de Kronecker[3] . On dit dans ce cas que $r(t)$ vérifie le critère de Nyquist.

Le critère de Nyquist permet de déterminer l'expression du filtre de réception pour qu'il soit adapté au filtre d'émission et à la forme d'onde. De plus on peut montrer qu'il existe une répartition optimale entre les deux filtres d'émission et de réception, que l'on nomme demi-Nyquist, ou racine de Nyquist. Pour une discussion plus détaillée sur le filtre de réception et la racine de Nyquist, le lecteur peut se référer à [GLAV96].

[3]Ce symbole est utilisé pour simplifier les notations. $\delta_{ij} = 1$ si $i = j$ et $\delta_{ij} = 0$ sinon.

1.7. Seuil de décision et décodage canal

L'étape suivante consiste à déterminer les bits correspondant au symbole reçu d_k après le filtre de réception. Ce symbole peut être différent du symbole qui avait été envoyé (c_k) à cause de perturbations introduites par le canal. La détection par maximum de vraisemblance est le critère optimal permettant de déterminer le symbole qui a été envoyé avec la plus grande probabilité. Pour cela on sélectionne le point de la constellation le plus proche (au sens de la distance euclidienne) du symbole reçu, et les bits qui sont associés à ce point de la constellation sont les bits qui ont été émis avec la plus grande vraisemblance. Le plan complexe est ainsi partitionné en zones de décision, chacune correspondant à un symbole de la constellation, et donc à un ensemble de bits particulier. Sur une constellation particulière, on peut représenter les limites de ces zones par des traits pointillés (on suppose que tous les symboles sont équiprobables) :

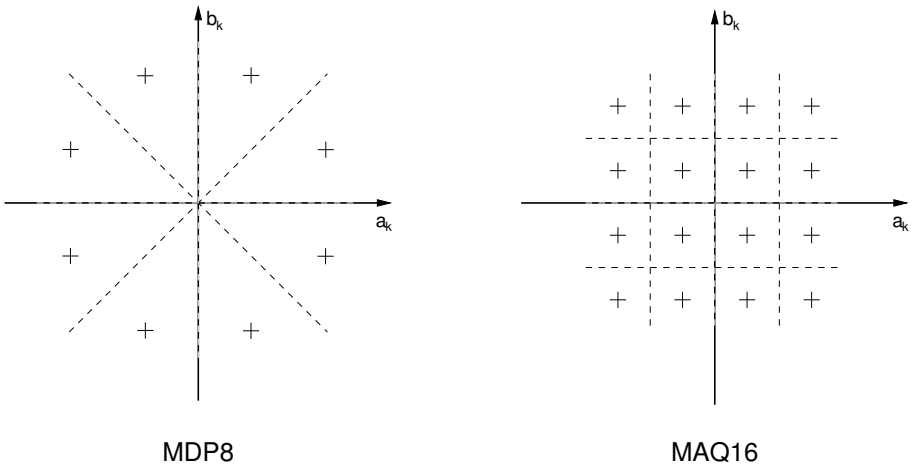

Figure 1.6. Frontières des zones de décision sur les constellations MDP8 et MAQ16

Le signal décidé (au sens du critère de maximum de vraisemblance), sous forme binaire, sera décode grâce au décodeur canal. Ce décodeur correspond au codeur canal qui a été utilisé dans l'émetteur pour ajouter de la redondance aux informations transmises (voir figure 1.1). Cette redondance est utilisée par le décodeur canal pour détecter des erreurs dans le flux binaire et éventuellement les corriger. Dans le cas d'un système FEC (Forward Error Correction) les erreurs sont corrigées directement par le décodeur, et dans le cas d'un système ARQ (Automatic Repeat reQuest) les erreurs sont seulement détectées et le système demande à l'émetteur de transmettre à nouveau les informations.

1.8. Cas du canal sélectif en fréquence

Jusqu'à présent on a supposé que le canal était parfait. Ceci n'est bien sûr pas le cas en pratique, et le canal déforme le signal transmis. Le modèle général est le canal multitrajets. Dans un canal radio, et particulièrement en milieu urbain, ou dans un environnement intérieur, les ondes peuvent être réfléchies par les obstacles environnants, et le signal reçu est une somme de différents échos atténués et retardés, qui ont chacun suivi un chemin différent.

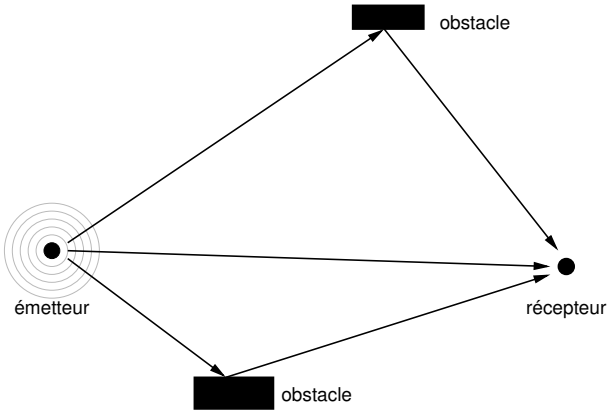

Figure 1.7. Milieu de transmission avec deux obstacles

Ce modèle est également retrouvé dans un canal filaire. Les ruptures d'impédance du câble (un raccord ou une dérivation par exemple) engendrent des réflexions du signal, ce qui cause également des échos avec différents retards et degrés d'atténuation.

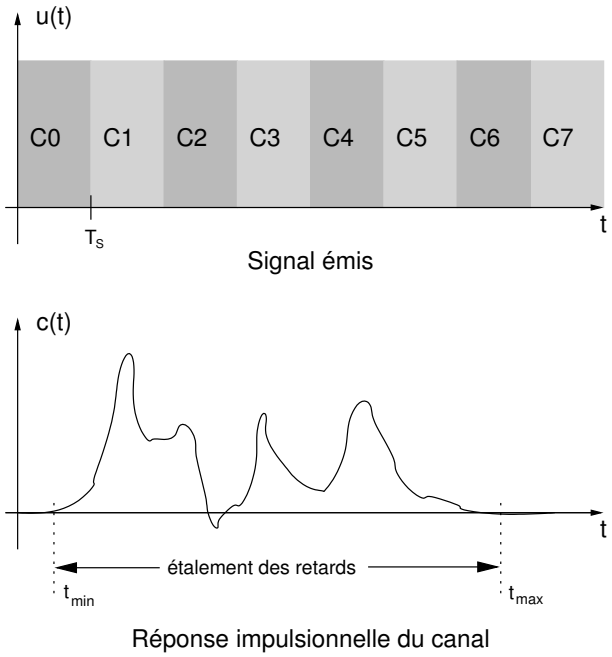

Figure 1.8. Interférences entre symboles ($t_{max} >> T_S$)

Si l'on appelle t_{min} le temps de propagation du trajet le plus court et t_{max} celui du trajet le plus long, on peut définir l'étalement des retards τ_R, qui est alors égal à $t_{max} - t_{min}$. Cet étalement correspond à l'intervalle de temps pendant lequel la réponse impulsionnelle du canal est non

nulle. En comparant l'étalement des retards avec la durée T_S d'un symbole, on peut déterminer s'il y aura une interférence entre symboles introduite par le canal. Si $T_S \gg \tau_R$, c'est-à-dire si la durée du symbole est grande devant l'étalement des retards, l'influence de ces retards est négligeable sur tout un symbole, et il y aura peu d'interférence. Si T_S est du même ordre de grandeur que τ_R, il y aura une interférence entre quelques symboles, et plus T_S est petit devant τ_R, plus le nombre de symboles qui interfèrent entre eux est grand.

Ce critère peut être exprimé également de manière fréquentielle, en comparant la bande passante Δf du signal modulé, et $1/\tau_R$. Si $\Delta f \ll 1/\tau_R$, les retards ne créeront pas d'interférence importante entre les symboles. Sinon plus Δf est grande devant $1/\tau_R$, plus l'interférence sera importante. On définit la bande de cohérence du canal B_c, qui est du même ordre de grandeur que $1/\tau_R$, et qui représente la largeur de bande de fréquence sur laquelle on considère que la réponse fréquentielle du canal varie peu, et donc la largeur de la bande que l'on peut utiliser sans avoir d'interférence significative. Généralement on définit la bande de cohérence à 3 dB, c'est à dire la bande de fréquences dans laquelle la réponse fréquentielle s'écarte de moins de 3 dB du maximum. Bien sûr la bande que l'on utilisera en pratique pourra être plus ou moins grande que la bande à 3 dB, en fonction de la tolérance voulue sur les perturbations introduites par les retards.

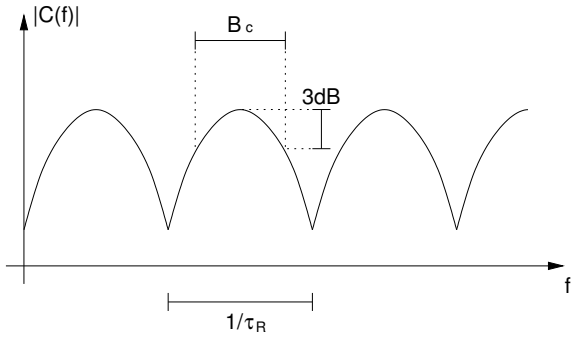

Figure 1.9. Exemple de réponse fréquentielle d'un canal à deux trajets, avec la bande de cohérence

Lorsque T_S est de l'ordre de grandeur ou inférieur à τ_R, il faut adapter le filtre de réception (équation (1.16)), et tenir compte de la réponse fréquentielle du canal. On intègre donc la réponse du canal $c(t)$ dans la fonction $r(t)$ devant vérifier le critère de Nyquist (équation (1.17)). Cette fonction est maintenant définie par $r(t) = (h \star g_e \star c \star g_r)(t)$. Si $h(t)$ et $g_e(t)$ sont connus, il n'en est pas de même de $c(t)$. En effet la réponse du canal dépend des conditions extérieures, et peut même varier au cours du temps. Plus T_S est petit devant τ_R (c'est à dire plus Δf est grand devant B_c), plus le problème de la détermination du $g_r(t)$ est délicat, car plus le nombre de symboles impliqués dans l'interférence est grand. Une méthode généralement employée est l'égalisation de canal, qui consiste à séparer le problème en deux. Le filtre de réception est décomposé en un filtre $g_r(t)$ adapté au filtre d'émission, et un second filtre $l(t)$ adapté au canal :

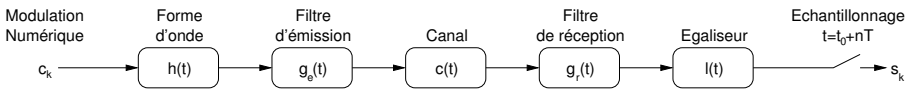

Figure 1.10. Filtre de réception et égaliseur

Le filtre de réception, $g_r(t)$, est déterminé de la même manière que dans le cas du canal parfait, tandis que le filtre $l(t)$ inverse la réponse du canal. Dans ce cas le critère de Nyquist est

également vérifié et la perturbation introduite par le canal est compensée. La difficulté de la méthode est l'évaluation de $c(t)$, qui n'est pas connue à priori, et peut évoluer dans le temps. C'est pourquoi on choisit en général un système adaptatif pour réaliser l'égaliseur. Il peut également être réalisé sous forme discrète, et dans ce cas il est placé après l'échantillonneur.

Les canaux multitrajets sont également appelé canaux sélectifs en fréquence, car comme on le voit sur la réponse fréquentielle figure 1.9, ils atténuent certaines bandes de fréquences et en amplifient d'autres. Les modulations multiporteuses sont une solution proposée pour s'adapter plus facilement à ces canaux difficiles.

2. Modulations multiporteuses

2.1. Principe

La modulation multiporteuses permet de simplifier le problème de l'égalisation dans le cas d'un canal sélectif en fréquence, c'est à dire lorsque l'étalement des retards τ_R est grand devant la durée d'un symbole T_S. Le principe est de transmettre simultanément plusieurs symboles en parallèle sur différences porteuses. En modulant sur N_p porteuses, il est possible d'utiliser des symboles N_p fois plus longs tout en conservant le même débit qu'avec une modulation monoporteuse. En choisissant une valeur assez grande pour N_p, la durée des symboles devient grande devant l'étalement des retards, et les perturbations liées aux échos deviennent négligeables.

La bande spectrale allouée à la transmission Δf est partagée entre les différentes porteuses, et ainsi chaque porteuse peut occuper une bande de fréquence inférieure à la bande de cohérence B_c du canal, même si Δf est grand devant la bande de cohérence. On peut remarquer qu'il existe une dualité temps-fréquence entre les modulations mono et multiporteuses. Une modulation monoporteuse réalise un multiplexage temporel, tandis qu'une modulation multiporteuses réalise un multiplexage fréquentiel, d'où le nom FDM (Frequency Division Multiplexing) :

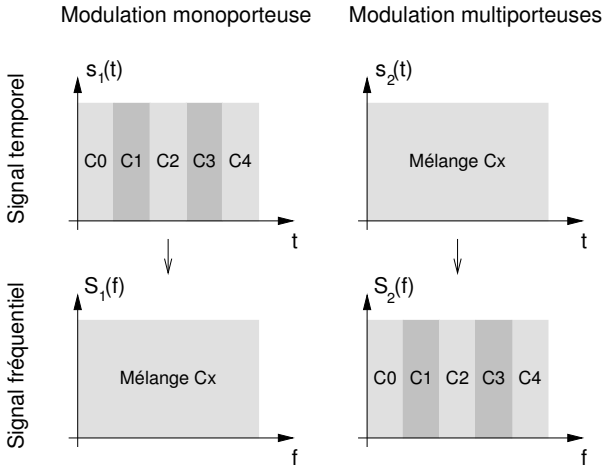

Figure 1.11. Dualité temps-fréquence des modulations

On définit l'efficacité spectrale comme étant le débit binaire par unité de fréquence. Plus l'efficacité spectrale est importante, plus il sera possible de transmettre un débit important sur un canal donné. Le choix des porteuses et de leur écartement va influer sur cette efficacité spectrale. Pour garder la même efficacité qu'avec la modulation monoporteuse équivalente, il faut choisir soigneusement les fréquences des porteuses utilisées. La méthode la plus répandue est l'utilisation de porteuses orthogonales.

2.2. Porteuses orthogonales

Pour que le signal modulé ait une grande efficacité spectrale, il faut que les fréquences des porteuses soient les plus proches possibles, tout en garantissant que le récepteur soit capable de les séparer et retrouver le symbole numérique émis sur chacune d'entre elles. Ceci est vérifié si le spectre d'une porteuse est nul aux fréquences des autres porteuses.

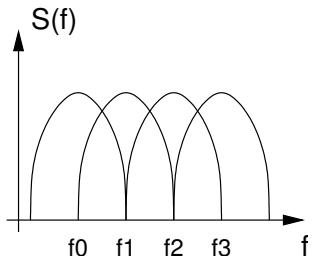

Figure 1.12. Porteuses espacées correctement pour une grande efficacité spectrale et une grande séparabilité

Le signal modulé sur une porteuse avec l'utilisation d'une forme d'onde rectangulaire (équations (1.5) et (1.6)), a un spectre défini par un sinus cardinal. En effet, en appelant T_S la durée d'un symbole et f_j la fréquence de la porteuse, son spectre S_j sera :

$$S_j(f) = T_S \frac{\sin(\pi(f - f_j)T_S)}{\pi(f - f_j)T_S} e^{-\pi i(f-f_j)T_S} \tag{1.18}$$

On remarque que ce spectre s'annule aux fréquences $f = f_j + k/T_S \ \forall k \neq 0$, donc $1/T_S$ est un espacement possible des fréquences des différentes porteuses. Ceci peut être vérifié graphiquement en superposant les spectres de plusieurs porteuses espacées de $1/T_S$:

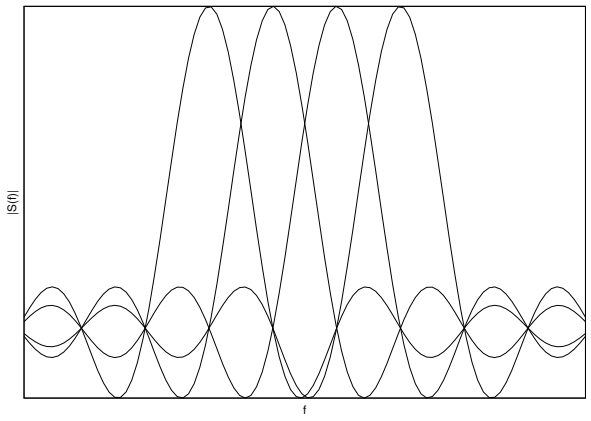

Figure 1.13. Superposition des spectres de 4 porteuses espacées de $1/T_S$

La largeur de bande du signal modulé est dans ce cas N_p/T_S, en appelant N_p le nombre de porteuses. Dans une modulation OFDM (Orthogonal Frequency Division Multiplexing), la base de signaux élémentaires (variante multiporteuses de la définition (1.12)) est la suivante [FLOC95] :

$$\psi_{j,k}(t) = e^{2i\pi(f_0 + \frac{k-N_p/2}{T_S})t} h(t - jT_S) \tag{1.19}$$

où f_0 est la fréquence centrale des porteuses et $h(t)$ la forme d'onde rectangulaire, définie en (1.6). Le signal modulé s'exprime alors sous la forme :

$$u(t) = \Re(\sum_{j=0}^{\infty} \sum_{k=0}^{N_p-1} c_{j,k}\psi_{j,k}(t)) \tag{1.20}$$

N_p symboles sont transmis en même temps, en parallèle, et le vecteur $\mathbf{c_j}$ de composantes $\{c_{j,k},\ k = 0 \dots N_p - 1,\ j \text{ fixé}\}$ est appelé symbole OFDM. Les symboles OFDM sont transmis successivement, et $c_{j,k}$ est le symbole transmis sur la $k^{\text{ième}}$ porteuse dans le $j^{\text{ième}}$ symbole OFDM. On remarque que la base de signaux est orthogonale : pour $j \neq j'$ et $k \neq k'$:

$$\begin{aligned} &\int_{-\infty}^{+\infty} \psi_{j,k}(t)\overline{\psi_{j',k'}(t)}\ \mathrm{d}t = 0 \\ &\int_{-\infty}^{+\infty} |\psi_{j,k}(t)|^2\ \mathrm{d}t = T_S \end{aligned} \tag{1.21}$$

Les porteuses sont orthogonales, d'où le O de l'abréviation OFDM donnée à cette modulation.

2.3. Réalisation et égalisation

La réalisation analogique d'un modulateur OFDM peut sembler complexe, puisqu'il faudrait en toute logique N_p modulateurs, bien synchronisés, et dont les fréquences sont espacées d'exactement $1/T_S$:

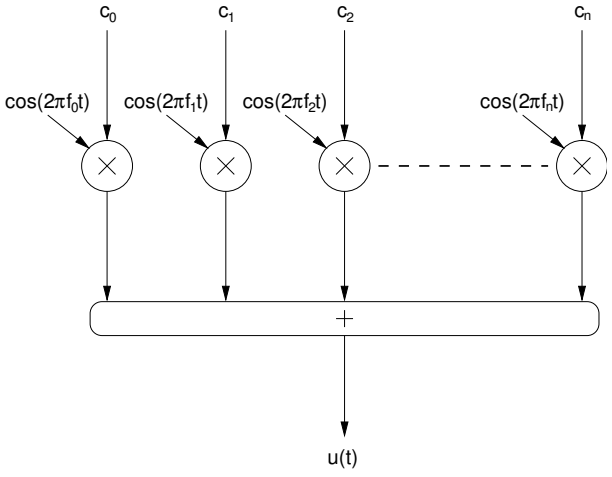

Figure 1.14. Réalisation possible d'un modulateur OFDM

Cependant, si l'on décompose l'expression d'un symbole OFDM :

$$\begin{aligned} u_j(t) &= \Re(\sum_{k=0}^{N_p-1} c_{j,k} e^{2i\pi(f_0+\frac{k-N_p/2}{T_S})t} h(t-jT_S)) \\ u_j(t) &= \Re(u_{ej}(t) e^{2i\pi f_0 t}) h(t-jT_S) \\ \text{avec } u_{ej}(t) &= \sum_{k=0}^{N_p-1} c_{j,k} e^{2i\pi \frac{k-N_p/2}{T_S} t} \end{aligned} \tag{1.22}$$

on reconnaît l'expression d'une transformée de Fourier inverse dans l'expression de $u_{ej}(t)$. Si l'on appelle $u_{j,l}$ les coefficients de l'IDFT (Inverse Discrete Fourier Transform) des $c_{j,k}$:

$$u_{j,l} = \sum_{k=0}^{N_p-1} c_{j,k} e^{2i\pi \frac{l(k-N_p/2)}{N_p}} \tag{1.23}$$

On peut construire un signal continu à partir des $u_{j,l},\ l = 0 \ldots N_p - 1$, cadencés à la fréquence N_p/T_S :

$$u'_{ej}(t) = \sum_{k=0}^{N_p-1} u_{j,k} h_{N_p}(t - jT_S - k\frac{N_p}{T_S}) \tag{1.24}$$

où $h_{N_p}(t)$ est la fonction rectangle de durée T_S/N_p. On constate que l'on peut obtenir le signal $u_{ej}(t)$ à partir de $u'_{ej}(t)$ et d'un filtre passe-bas avec une fréquence de coupure de N_p/T_S. On propose donc d'utiliser le signal suivant :

$$u'_j(t) = \Re(\sum_{k=0}^{N_p-1} u_{j,k} e^{2i\pi f_0 t} h_{N_p}(t - jT_S - k\frac{T_S}{N_p})) \tag{1.25}$$

Après un filtrage autour de la fréquence f_0 avec une largeur de bande de N_p/T_S, le signal obtenu sera le même qu'en utilisant celui défini en (1.22). Or cette expression correspond à la modulation d'amplitude en quadrature classique des coefficients $u_{j,k}$ à la fréquence f_0. On peut donc réaliser un émetteur OFDM en utilisant une IDFT et un seul modulateur.

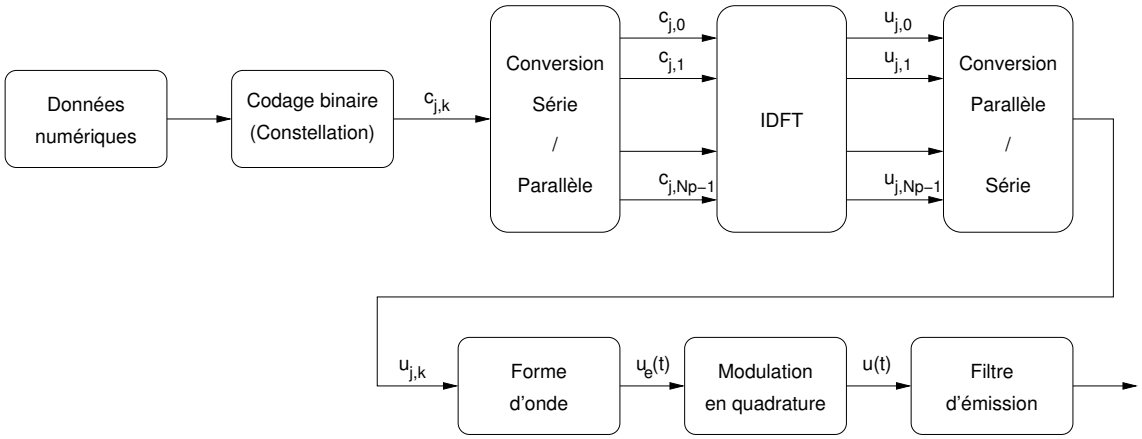

Figure 1.15. Réalisation d'un émetteur OFDM avec une IDFT

Pour limiter les distorsions dues au canal on choisit le nombre de porteuses N_p de telle sorte que la durée T_S d'un symbole OFDM soit longue devant celle de la réponse impulsionnelle du canal $c(t)$. Cette condition peut se traduire dans le domaine fréquentiel : on cherche à avoir une largeur de bande d'une porteuse $1/T_S$ petite devant la bande de cohérence du canal. Ainsi on peut considérer que la réponse fréquentielle du canal est constante pour chaque porteuse :

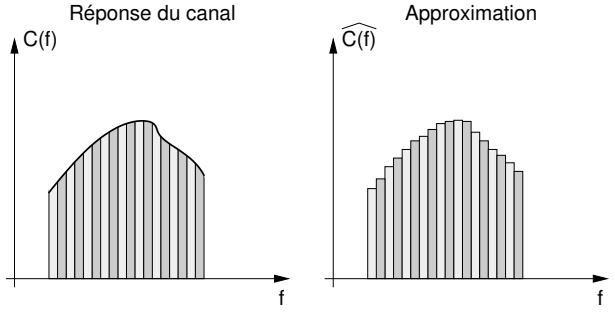

Figure 1.16. Réponse fréquentielle du canal

Les filtres d'émission et de réception ($G_e(f)$ et $G_r(f)$) sont des filtres passe-bande qui sont conçus pour ne pas déformer le signal transmis dans sa bande utile. On peut donc dire que la réponse fréquentielle de la chaîne de transmission complète $R(f) = H(f)G_e(f)C(f)G_r(f)$ aura elle aussi des variations très faibles dans la bande de fréquence de chaque porteuse. Ainsi le signal reçu par le récepteur après filtrage pour le symbole OFDM numéro j peut être représenté par :

$$\begin{aligned} v_j(t) &= \sum_{k=0}^{N_p-1} e^{2i\pi(f_0+\frac{k-N_p/2}{T_S})t} r_k c_{j,k} h(t-jT_S) \\ \text{avec } r_k &= R(f_0 + \frac{k-N_p/2}{T_S}) \end{aligned} \tag{1.26}$$

r_k est la valeur de $R(f)$ à la fréquence f_k. Le signal est ensuite démodulé (à l'aide d'une porteuse à la fréquence f_0) puis échantillonné à la fréquence N_p/T_S, et les N_p symboles suivants sont obtenus :

$$v_{j,l} = \sum_{k=0}^{N_p-1} e^{2i\pi(k-N_p/2)\frac{l}{N_p}} r_k c_{j,k} \tag{1.27}$$

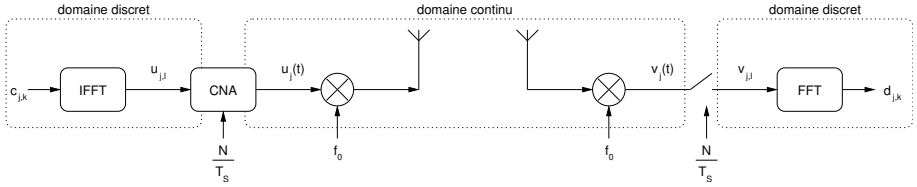

Figure 1.17. Utilisation des transformées de Fourier discrètes dans un système OFDM

On remarque qu'en réalisant une transformée de Fourier discrète (DFT, pour Discrete Fourier Transform) des coefficients $v_{j,l},\ l = 0 \ldots N_p - 1$, on obtient les symboles $d_{j,k} = r_k c_{j,k},\ k = 0 \ldots N_p - 1$. Ainsi il suffit pour le récepteur de diviser les symboles reçus par r_k pour retrouver les symboles émis. L'égalisation est donc beaucoup plus simple à réaliser qu'avec une modulation monoporteuse. Les r_k ne sont pas connus à priori, mais il est très simple de les évaluer en insérant des symboles OFDM de calibrage, connus de l'émetteur et du récepteur. Une réalisation possible d'un récepteur OFDM est donc :

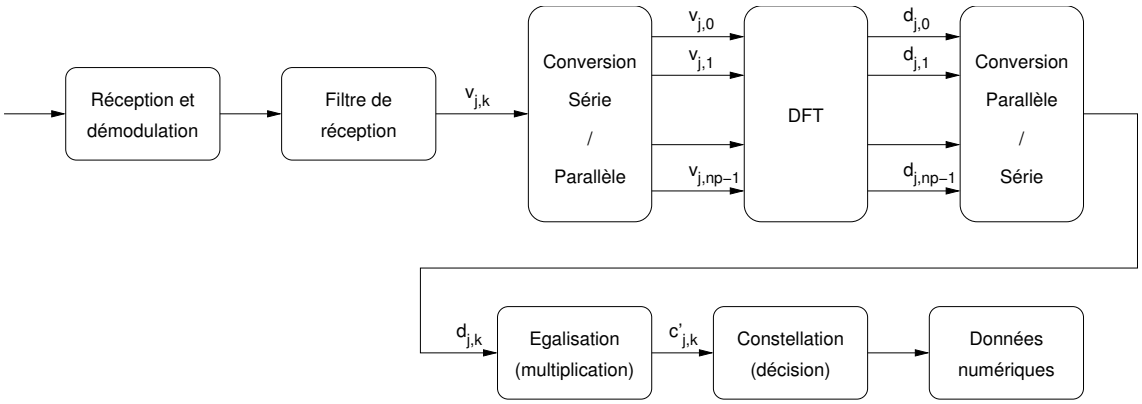

Figure 1.18. Réalisation d'un récepteur OFDM avec une DFT

2.4. Intervalle de garde

Une multiplication des coefficients reçus suffit donc à compenser les distorsions du canal au sein d'un symbole OFDM. Cependant il peut subsister une légère interférence entre deux symboles OFDM transmis successivement. Pour s'en affranchir, il est possible d'ajouter un espace entre les symboles OFDM, d'une durée supérieure à l'étalement des retards. Ainsi les derniers échos du symbole OFDM auront lieu durant cet intervalle dit "de garde", et le symbole OFDM suivant ne sera plus perturbé par le précédent. En pratique on choisit pour la taille de cet intervalle de garde une durée de l'ordre du quart de celle d'un symbole OFDM, ce qui représente un bon compromis entre diminution des erreurs et perte de débit utile.

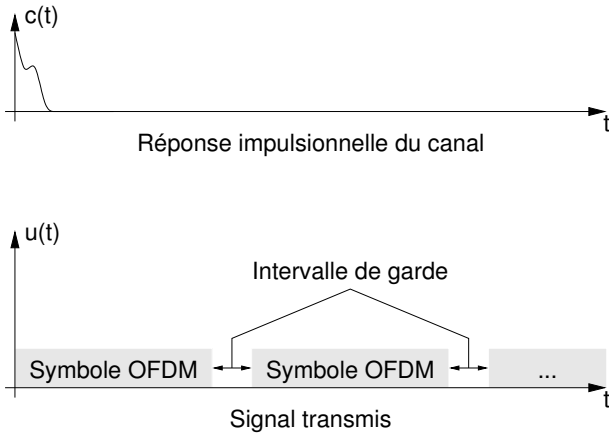

Figure 1.19. Intervalle de garde

Plusieurs mécanismes sont donc présents dans une transmission OFDM pour réduire les erreurs. L'intervalle de garde réduit les interférences entre deux symboles OFDM, dues aux trajets multiples. L'égalisation du canal que l'on réalise avec une simple multiplication réduit les interférences à l'intérieur d'un symbole OFDM, puis le code canal corrige les erreurs supplémentaires, dues principalement au bruit. L'utilisation d'un code correcteur d'erreurs avec une modulation OFDM est appelée COFDM (Coded OFDM).

3. OFDM et non-linéarités

3.1. Facteur de crête et éléments non linéaires

Dans une modulation monoporteuse, l'ensemble des valeurs prises par le signal modulé est fixé par les constellations, et la distribution des valeurs prises suit généralement une loi uniforme.

Le signal OFDM temporel quand à lui est la somme de N_p signaux modulés, où N_p est le nombre de porteuses. D'après le théorème de la limite centrale, si $N_p \rightarrow \infty$, la distribution des valeurs prises par le signal OFDM temporel tend vers une variable aléatoire normale. En pratique, on peut constater par simulation que la distribution des valeurs prises par $u_e(t)$ est proche d'une loi gaussienne à partir de 4 porteuses.

Si l'on compare un signal à distribution uniforme et un signal à distribution gaussienne de même puissance, on constate des différences. Le signal gaussien a une dynamique plus grande, et on peut remarquer la présence de 'pics' d'amplitude importante. Dans la figure 1.20 suivante,

les deux signaux ont une variance de 1. Le signal du haut est un signal à répartition gaussienne, et le signal du bas est un signal à répartition uniforme :

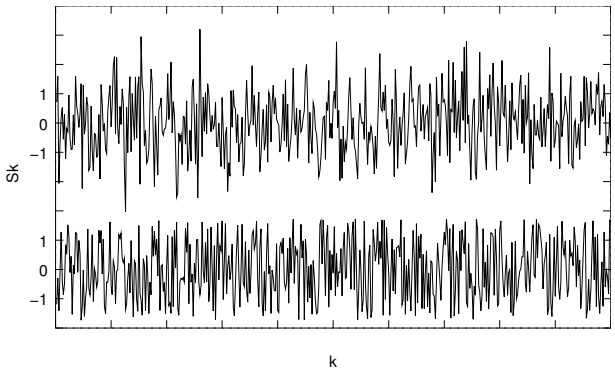

Figure 1.20. Signaux gaussien (haut) et uniforme (bas)

On peut quantifier cette caractéristique avec une grandeur appelée facteur de crête. On définit tout d'abord le PMEPR (Peak-to-Mean Envelope Power Ratio), qui est le rapport entre la puissance maximale et la puissance moyenne d'un signal temporel. Si un signal non nul $s(t)$ et de moyenne nulle est défini sur $[0, T]$, son PMEPR sur cet intervalle est égal à :

$$PMEPR(s) = \frac{\max\limits_{t\in[0,T]} |s(t)|^2}{\dfrac{1}{T}\displaystyle\int_0^T |s(t)|^2\, dt} \tag{1.28}$$

Le facteur de crête (CF, pour Crest Factor) est défini lui par $CF = \sqrt{PMEPR}$. Un facteur de crête élevé signifie que le signal possède une puissance maximale importante devant sa puissance moyenne, et donc que certaines valeurs prises par ce signal sont importantes par rapport aux valeurs moyennes. Autrement dit ceci signifie que des pics d'amplitude importante sont présents.

Dans le cas d'un signal $s_A(t)$ à distribution uniforme centrée d'amplitude A, la puissance maximale est égale à A^2 (car le signal prend des valeurs entre $-A$ et A) et on peut calculer sa puissance moyenne, qui est égale à $A^2/3$. Le PMEPR d'un signal uniforme est donc égal à 3. Pour un signal centré à distribution normale de variance σ^2, la puissance maximale est infinie (car il n'y a aucune limite sur les valeurs prises par le signal) et la puissance moyenne est égale à σ^2. Le PMEPR est donc infini.

Dans le cas d'un signal OFDM on montre [LOUE00] que le PMEPR est majoré par :

$$PMEPR \leq N_p \frac{\max\limits_{j,k} |c_{j,k}|^2}{\min\limits_{j,k} |c_{j,k}|^2} \tag{1.29}$$

où N_p est le nombre de porteuses et $c_{j,k}$ sont les symboles modulés sur chaque porteuse. On montre également que le PMEPR est exactement égal à N_p dans le cas d'une modulation de phase (c'est-à-dire lorsque tous les symboles c_k ont le même module). Plus on augmente N_p, plus le PMEPR est élevé. Comme la modulation OFDM devient avantageuse lorsque ce

nombre de porteuses est grand, dans la plupart des standards mettant en application l'OFDM le PMEPR du signal temporel sera élevé.

L'évaluation du facteur de crête est important pour le dimensionnement des composants non linéaires dans un système de communication. Un amplificateur est nécessaire afin de faire se propager le signa sur le canal. Si la transmission est réalisée sur un canal radio, on utilise un amplificateur de puissance afin que l'onde radio ait une puissance suffisante. Or les amplificateurs radio utilisés en pratique, à semi-conducteurs, ou SSPA (Solid-State Power Amplifier) ont une caractéristique non linéaire. La courbe ci-dessous représente la courbe de réponse d'un amplificateur SSPA typique :

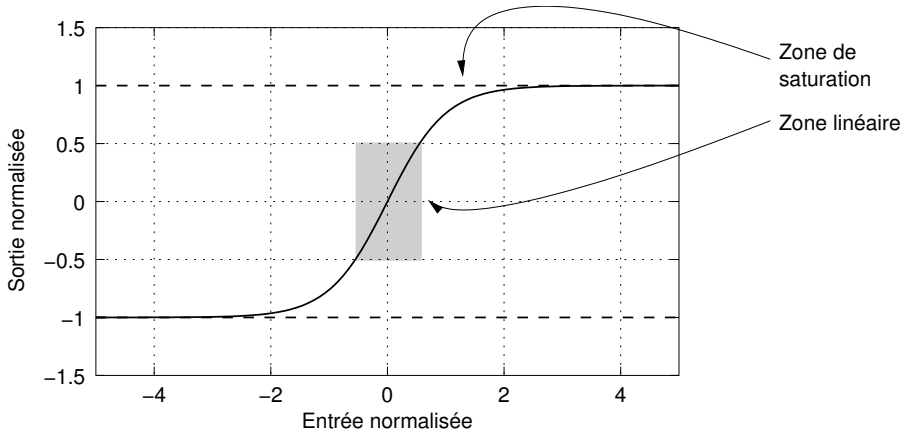

Figure 1.21. Caractéristique typique d'un amplificateur SSPA (échelle linéaire)

Dans la courbe de réponse de l'amplificateur, la sortie est normalisée de telle manière que la puissance de saturation corresponde à 0 dB (amplitude 1), et l'entrée de telle manière que le gain dans la zone linéaire soit de 1.

Sur une certaine plage de valeurs de l'entrée, l'amplificateur a un comportement très proche d'un système linéaire, et le signal en sortie sera tout simplement proportionnel au signal d'entrée, selon un rapport appelé gain de l'amplificateur. Cette plage est appelée zone de fonctionnement linéaire de l'amplificateur, et est indiquée en gris sur la caractéristique ci-dessus. Pour une amplitude plus grande de l'entrée la réponse de l'amplificateur s'éloigne de celle d'un système linéaire, et la sortie tend vers une valeur limite, appelée saturation en sortie. On peut définir la taille de la zone linéaire en utilisant la notion de point de compression à 1 dB, qui est le point de la caractéristique qui s'éloigne d'1 dB de la loi linéaire :

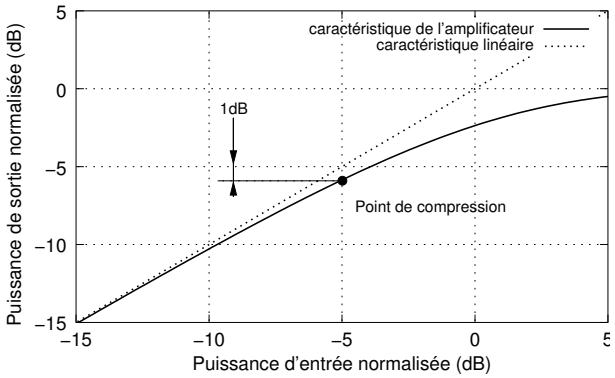

Figure 1.22. Point de compression à 1 dB

Pour que le signal ne subisse aucune distorsion dans l'amplificateur, il est nécessaire que celui-ci reste dans la zone de fonctionnement linéaire, et donc que sa puissance maximale soit inférieure à celle correspondant au point de compression. Si le facteur de crête est élevé, il sera nécessaire de surdimensionner l'amplificateur, c'est à dire de le choisir de telle sorte que la puissance de compression soit largement supérieure à la puissance moyenne du signal, le rapport entre les deux puissances étant égal au PMEPR. Donc pour deux signaux de puissance moyenne égale, celui qui a le PMEPR le plus élevé demandera un amplificateur avec une puissance de saturation plus élevée. Mais plus on augmente la puissance de saturation de l'amplificateur, plus on augmente son coût et sa consommation énergétique. Ainsi lorsque le signal transmis possède un facteur de crête élevé il est souvent nécessaire de trouver un compromis entre puissance de l'amplificateur et distorsion, et donc des perturbations dûes à la non-linéarité vont apparaître.

On définit une autre grandeur qui représente l'influence de la non-linéarité de l'amplificateur sur un signal donné, appelée recul d'entrée, ou Input Back-Off (IBO) en anglais. Cette grandeur, généralement exprimée en dB, est le rapport entre la puissance de saturation ramenée à l'entrée[4] de l'amplificateur et la puissance moyenne du signal. Plus le recul d'entrée est élevé, plus l'amplificateur est surdimensionné par rapport au signal à amplifier, et moins il y a de distorsions non linéaires.

Dans le cas d'une liaison filaire, l'amplificateur n'a pas une puissance aussi importante, mais l'amplitude du signal temporel est tout de même limitée par la dynamique du convertisseur numérique/analogique, et souvent la puissance du signal émis doit être limitée pour éviter un rayonnement trop important. Le modèle de composant non-linéaire choisi dans ce cas est un amplificateur linéaire à saturation, ou clipping en Anglais. Dans le reste de ce mémoire, un amplificateur ayant une telle caractéristique sera appelé limiteur.

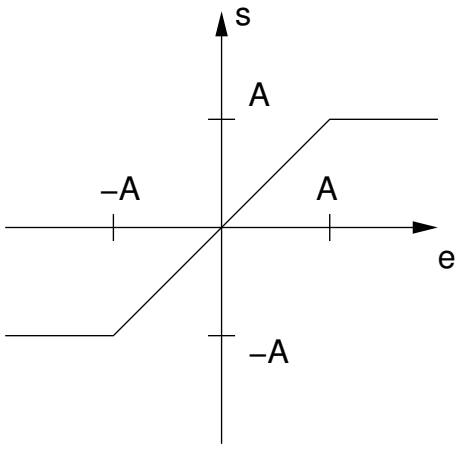

Figure 1.23. Caractéristique d'un limiteur

$$s = \begin{cases} A & \text{si } e \geq A \\ e & \text{si } -A < e < A \\ -A & \text{si } e \leq -A \end{cases} \tag{1.30}$$

[4] c'est à dire la puissance de saturation en sortie divisée par le gain en puissance de l'amplificateur

3.2. Harmoniques et intermodulations

Les dégradations véhiculées par les non linéarités peuvent être observées dans le domaine fréquentiel. Si l'on fournit à l'entrée d'un composant non linéaire un signal sinusoïdal de fréquence f_0, le signal disponible en sortie n'est plus sinusoïdal, mais il reste périodique de période $t_0 = 1/f_0$. D'après le théorème de Fourier, le signal de sortie peut donc être décomposé en une somme de sinusoïdes de fréquences $kf_0,\ k = 0 \ldots \infty$. Sur le spectre du signal de sortie, on voit donc apparaître des raies supplémentaires, appelés harmoniques, qui ont des fréquences multiples de la fréquence d'origine f_0.

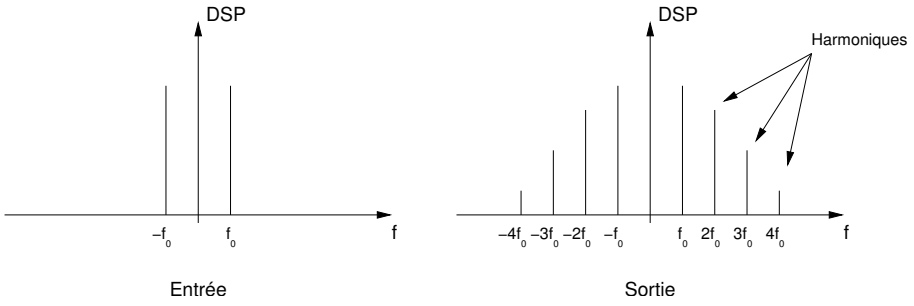

Figure 1.24. Harmoniques à la sortie d'un composant non linéaire

Dans le cas d'un signal radio, sa largeur de bande Δf est faible devant sa fréquence centrale f_0. Donc les signaux harmoniques se situent en dehors de la bande utile du signal radio. Pour respecter les contraintes du canal radio il est nécessaire de filtrer les harmoniques après l'amplificateur, afin d'éviter de perturber les canaux radio aux fréquences $kf_0, k\rangle 1$, mais les harmoniques ne perturbent pas le signal dans la bande utile $[f_0 - \Delta f/2, f_0 + \Delta f/2]$.

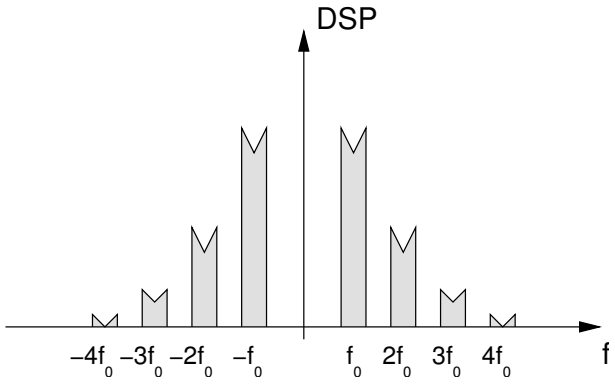

Figure 1.25. Harmoniques sur un signal en bande étroite

Le second effet fréquentiel des non-linéarités est l'intermodulation, qui apparaît lorsque plusieurs signaux à des fréquences différentes traversent un composant non linéaire. En effet si un signal composé de deux sinusoïdes aux fréquences f_1 et f_2 est fourni en entrée d'un composant non linéaire, le signal présent à la sortie comporte en plus des deux sinusoïdes d'autres signaux dont les fréquences sont des combinaisons linéaires de f_1 et f_2. Pour un signal radio à bande étroite, on ne s'intéresse qu'aux signaux présents dans la bande utile, et donc aux signaux à des fréquences voisines de f_1 et f_2. On appelle intermodulations d'ordre 3 les signaux présents au fréquences $2f_1 - f_2$ et $2f_2 - f_1$, intermodulations d'ordre 5 ceux aux fréquences $3f_1 - 2f_2$ et $3f_2 - 2f_1$, et ainsi de suite. Si on appelle δf l'espacement entre les

deux signaux d'origine ($\delta f = f_2 - f_1$) on remarque que les intermodulations sont également espacées de δf.

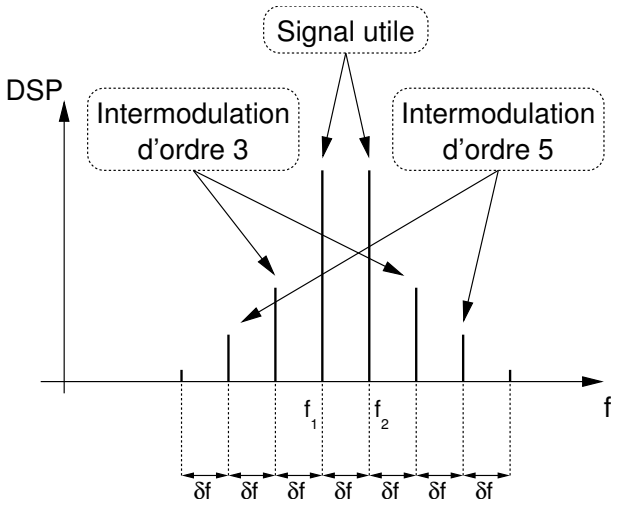

Figure 1.26. Intermodulations

3.3. Conséquences sur l'OFDM

Un signal OFDM temporel ayant un facteur de crête fluctuant, celui-ci peut subir des distorsions dues aux non-linéarités de l'amplificateur. Ces distorsions sont principalement des intermodulations. Les différentes porteuses sont régulièrement espacées (l'intervalle étant $1/T_S$) et donc la plupart des intermodulations dues à l'interférence de deux porteuses se trouveront à la fréquence d'une autre porteuse. Ainsi un symbole $d_{j,k}$ reçu par le récepteur OFDM n'est plus simplement de la forme $r_k c_{j,k}$, mais est une combinaison non linéaire de tous les symboles $c_{j,k}$ $j = 1 \ldots N_p$ qui composent le symbole OFDM. Au premier abord on peut considérer que la distorsion non linéaire est un bruit supplémentaire ajouté au symbole transmis, et si l'amplitude de ce bruit est supérieure au seuil de décision du récepteur, des erreurs binaires supplémentaires sont présentes dans le flux binaire reçu.

La figure 1.27 représente l'évolution du taux d'erreur binaire en fonction du rapport signal/bruit en utilisant des amplificateurs avec différents reculs. Le système OFDM simulé utilise 16 porteuses, une modulation de type MAQ16, et un modèle d'amplificateur SSPA.

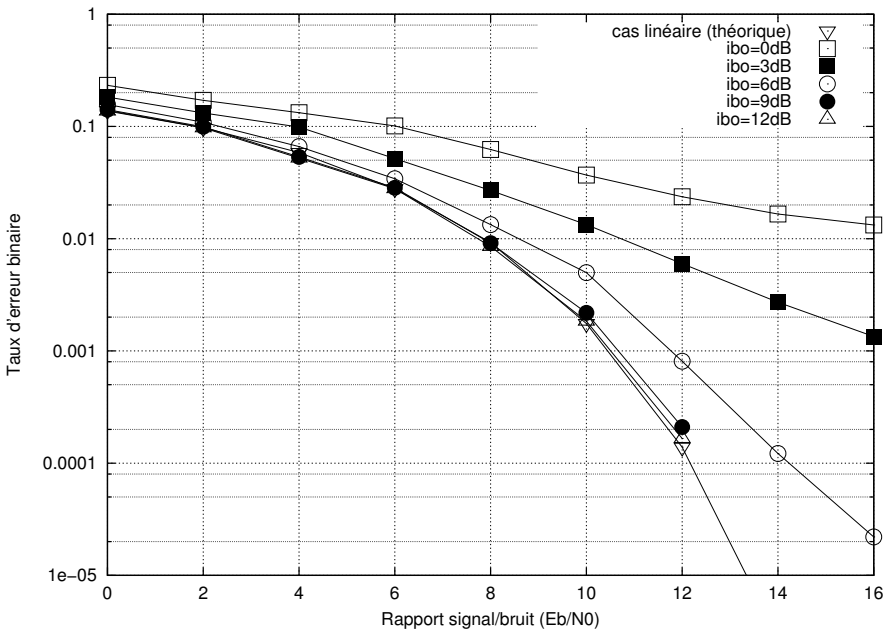

Figure 1.27. Taux d'erreur binaire en fonction du recul et du rapport signal sur bruit

Le canal simulé est un canal gaussien, c'est à dire qu'on ajoute au signal transmis un bruit blanc additif gaussien (BBAG). Le rapport signal sur bruit, exprimé en dB et noté E_b/N_0, est le rapport entre l'énergie consommée pour transmettre un bit et la densité spectrale de puissance du bruit gaussien. Pour des faibles rapports signal sur bruit, le bruit gaussien est la perturbation principale du signal et le taux d'erreur binaire ne dépend pas beaucoup du recul. Par contre quand E_b/N_0 est plus grand, le taux d'erreur binaire (TEB) chute quand le recul est assez élevé, mais stagne dans le cas contraire. A partir d'un certain recul on ne remarque plus de perturbation liées aux non linéarités. La courbe avec un recul de 12 dB est proche des résultats théoriques obtenus dans un cas linéaire.

De nombreuses études portent donc sur les non-linéarités en OFDM, aussi bien théoriques ([SHI96] pour le cas général, et [GROS93] pour celui de l'ADSL) que pratiques [MERC98].

4. La réduction des effets des non-linéarités en OFDM

Plusieurs méthodes ont déjà été proposées pour limiter les effets des non-linéarités dans un système OFDM. Celles-ci peuvent être classées en plusieurs catégories.

4.1. Limitation de l'amplitude du signal OFDM temporel

M. Pauli et H.-P. Kuchenbacker proposent de limiter l'amplitude du signal temporel en multipliant ce dernier par une enveloppe [[PAUL96]]. On fixe l'amplitude maximale voulue pour le signal OFDM, et un algorithme construit l'enveloppe à l'aide d'impulsions gaussiennes de telle sorte que ce seuil ne soit jamais dépassé. Les impulsions gaussiennes ont l'avantage d'être localisées à la fois dans les domaines fréquentiel et temporel, ce qui minimise les perturbations générées.

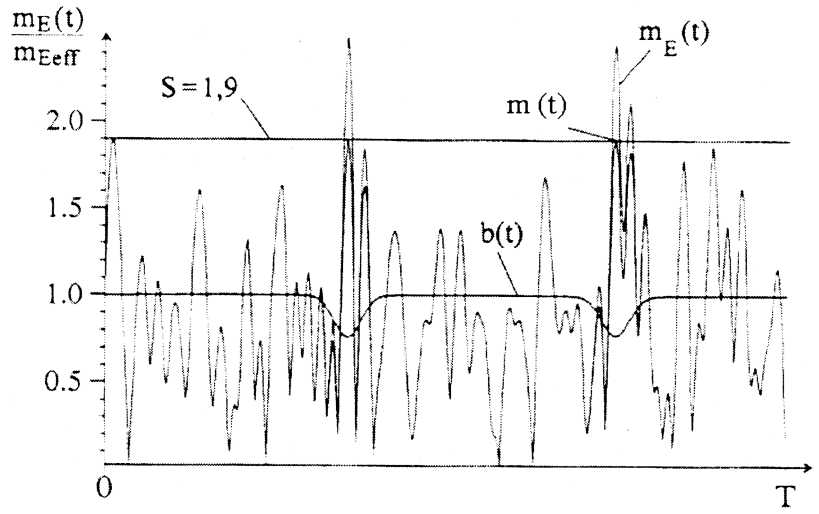

Figure 1.28. Impulsions gaussiennes pour limiter l'amplitude du signal

Dans la figure 1.28, extraite de l'article [PAUL96], $m_E(t)$ est le signal d'origine (sur la courbe il est divisé par sa valeur efficace m_{Eeff}), et il est multiplié par $b(t)$ composé d'impulsions gaussiennes. Le résultat du produit est le signal $m(t)$, qui ne dépasse jamais le seuil fixé, ici $S = 1,9$.

Le signal ayant une amplitude plus faible, les influences des non-linéarités de l'amplificateur seront moindres. Le but de cette méthode n'est pas de réduire le nombre d'erreurs, mais de diminuer les intermodulations crées par l'amplificateur quand il est utilisé dans son domaine non-linéaire. En effet dans certaines applications de l'OFDM, comme la norme de réseau informatique radio HIPERLAN/2, le domaine radio alloué est séparé en plusieurs canaux très proches, et le signal OFDM doit être transmis dans un seul de ces canaux. Cependant les intermodulations peuvent causer des perturbations dans un canal contigu, et nécessitent donc d'être contrôlées. Avec cette méthode les intermodulations sont fortement atténuées, avec une augmentation négligeable des erreurs sur le flux binaire.

4.2. Modification du codage

Un codage détecteur ou correcteur d'erreurs est systématiquement utilisé dans un système de communication numérique, afin de protéger la transmission en réduisant les erreurs binaires. Les codes en blocs sont particulièrement adaptés à une communication multiporteuses. Dans un code en bloc on encode un mot de m bits (appelé mot d'information) en un autre mot de n bits (appelé mot de code). $n \rangle m$ et $n - m$ est le nombre de bits de redondance par mot du code. Si l'on choisit pour n le nombre de bits représentés par un symbole OFDM, on a une correspondance directe entre un mot du code et un symbole OFDM. Or tous les symboles OFDM possibles n'ont pas le même PMEPR. Donc il est possible de sélectionner les mot de code qui génèrent des symboles OFDM avec un PMEPR faible, et ainsi diminuer le PMEPR global du signal temporel.

Il est possible de construire un nouveau code correcteur d'erreurs en choisissant systématiquement les mots de code ayant le PMEPR le plus faible. Cependant il n'a pas encore été trouvé d'algorithme permettant de déterminer ces mots, et il est donc nécessaire de calculer le PMEPR de tous les mots possibles avant de sélectionner ceux qui ont le facteur de crête le plus faible.

Quand le nombre de porteuses est important, le grand nombre de symboles possibles rend cette recherche très longue. Cependant on constate une chute importante du PMEPR avec des codes qui ont un bon rendement. Le décodage de ce type de codes est aussi extrêmement coûteux, car il faut comparer le mot reçu avec tous les mots du code [WILK95].

Des recherches ont dont été orientées vers des codes qui seraient à la fois simples à encoder et décoder et présentant un faible facteur de crête. Une famille de codes, les m-sequences, a été proposée dans [LI97], mais n'offre que peu d'avantages par rapport à d'autres familles [JEDW97]. Un autre type de code proposé est un code de Reed Müller d'ordre 1 associé à des séquences complémentaires de Golay. Il n'est plus nécessaire de parcourir toute la liste des symboles OFDM possibles, car un algorithme permet de déterminer directement un mot de code qui génère un signal avec un faible facteur de crête. De plus le décodeur est plus simple à réaliser. Le taux d'erreur binaire sur une liaison OFDM a ainsi été réduit de façon significative, mais cette technique est limitée aux modulations de phase (et donc aux modulations de type MDP à n états) [DAVI97][LOUE00]. Une famille particulière des codes de Reed-Solomon, nommée "Analog Codes" est également intéressante, mais nécessite que les filtres d'émission soient placés après la non-linéarité, ce qui exclut les applications en radiofréquence [HENK00].

4.3. Modification des symboles OFDM

La méthode PTS (Partial Transmit Sequences) [MULL97] consiste à répartir les différentes porteuses en plusieurs blocs, puis en appliquant une rotation sur chaque bloc. Le symbole OFDM $\mathbf{c_j}$ est tout d'abord décomposé en une somme. Si l'on appelle $\mathbf{c_j^{(l)}}$ chaque terme et si l'on note N_s le nombre de termes, on a :

$$\mathbf{c_j} = \sum_{l=1}^{N_s} \mathbf{c_j^{(l)}} \tag{1.31}$$

Sur chaque porteuse, le symbole $c_{j,k}$ transmis est placé dans un et un seul terme de la décomposition. Pour les autres termes, la composante correspondante sera à 0. On peut formaliser cette contrainte :

$$\forall k \in \{0 \dots N_p - 1\}\ \exists l \in \{1 \dots N_s\} \text{ tel que } c_{j,k}^{(l')} = \begin{cases} c_{j,k} & \text{si } l' = l \\ 0 & \text{si } l' \neq l \end{cases} \tag{1.32}$$

A titre d'exemple, on peut envisager une décomposition en deux termes ($N_s = 2$) dans laquelle le premier terme $c_{j,k}^{(1)}$ contiendrait les $N_p/2$ premières porteuses, et le symbole $c_{j,k}^{(2)}$ les $N_p/2$ suivantes. Ou encore un premier terme avec les porteuses de numéro pair, et un autre avec celles de numéro impair.

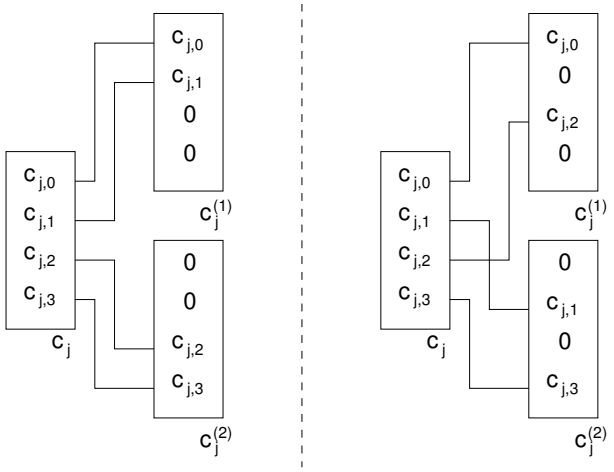

Figure 1.29. Exemples de décompositions PTS

Dans un modulateur OFDM (1.15), une IDFT est réalisée pour obtenir le symbole OFDM temporel. Comme l'IDFT est linéaire, il est possible de calculer l'IDFT de chaque symbole $\mathbf{c_j^{(l)}}$, et la somme des transformées sera également le symbole OFDM temporel. Dans un modulateur PTS, chaque transformée est multipliée par une valeur complexe α_k de module 1, ce qui cause une rotation du vecteur transformé. Un algorithme détermine le jeu de valeurs qui produit un symbole temporel avec le PMEPR le plus faible. Le récepteur réalisera la même recomposition, et multipliera par l'inverse des valeurs α_k, et ainsi retrouvera le symbole OFDM original.

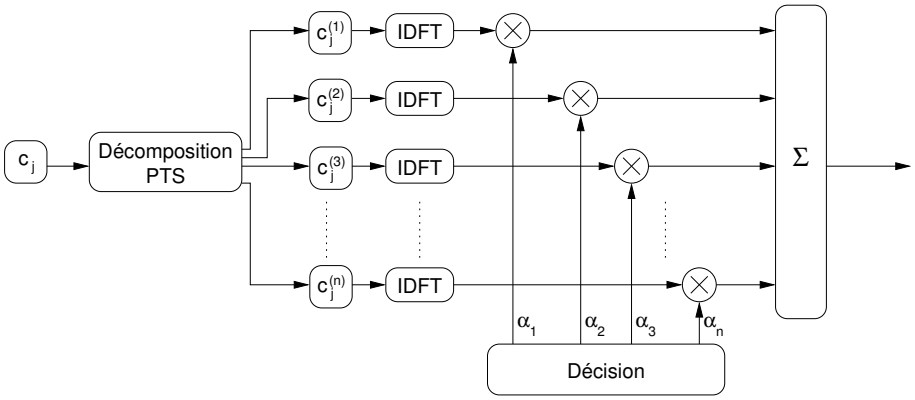

Figure 1.30. Principe d'un modulateur PTS

Le récepteur doit connaître les coefficients α_k pour recombiner les vecteurs correctement. Deux solutions sont proposée dans [MULL97]. On peut coder ces coefficients dans les symboles transmis, ou utiliser un codage différentiel. Dans les deux cas on constate une petite perte du débit utile, consacrée à ces coefficients. Cependant une baisse conséquente du PMEPR apparaît, lorsque le nombre de termes et la décomposition sont bien choisis.

Une autre méthode, dite "selected mapping" [BAUM96], peut être vue comme un cas particulier de PTS, où chaque terme $\mathbf{c_j^{(l)}}$ correspond à une et une seule porteuse. Dans un système "selected mapping", un certain nombre N_v de vecteurs $\mathbf{p_l}$ sont définis et connus de l'émetteur et du récepteur. Ces vecteurs ont N_p composantes, toutes complexes de module 1. Pour chaque symbole OFDM $\mathbf{c_j}$ on peut construire N_v symboles modifiés en multipliant terme à terme les vecteurs $\mathbf{c_j}$ et $\mathbf{p_l}$. C'est le symbole modifié qui génère un signal temporel avec

le PMEPR le plus faible qui sera transmis. Si le récepteur connaît le vecteur $\mathbf{p_l}$ qui a servi à modifier le symbole OFDM, il sera capable de retrouver le symbole original. Pour cela soit l'on transmet le numéro l du vecteur qui a fait la modification (au détriment d'une petite partie du débit utile), soit le récepteur effectue N_v décodages avec tous les vecteurs $\mathbf{p_l}$ possibles et détermine le symbole qui a été envoyé avec la plus forte probabilité.

Il est également possible d'ajouter aux symboles OFDM transmis une séquence particulière qui va baisser le facteur de crête. Les séquences possibles font partie d'un code connu de l'émetteur et du récepteur, et un algorithme en treillis permet de choisir la séquence qui entraîne un facteur de crête faible. Cette méthode est appelée "trellis shaping" [HENK00b].

Enfin deux méthodes décrites dans [TELL98] proposent d'agir sur les symboles transmis sur chaque porteuse. Dans la première méthode, dite "tone reservation", certaines porteuses sont réservées et ne transmettent aucune information. Un algorithme détermine quels symboles transmettre sur ces porteuses pour minimiser le facteur de crête, mais le récepteur les ignore. On constate une baisse importante du facteur de crête, au détriment du débit utile d'informations. Dans la seconde méthode, dite "tone injection", la constellation utilisée lors du codage binaire est étendue. Des points sont ajoutés autour de la constellation existante, et ainsi chaque symbole possible correspond à plusieurs points de la constellation. Un algorithme dans l'émetteur choisit les points qui minimisent le facteur de crête. Dans ce cas également le facteur de crête peut être réduit significativement, mais cette fois au détriment de la puissance moyenne du signal, qui augmente.

4.4. Prédistorsion

La prédistorsion consiste à ajouter un élément avant l'amplificateur qui va inverser la fonction non linéaire de celui-ci.

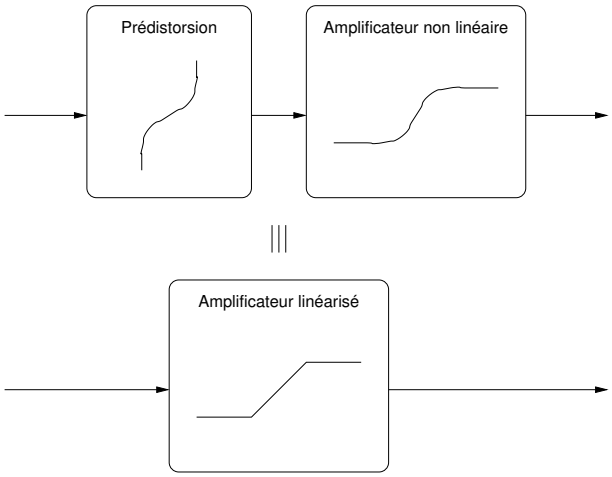

Figure 1.31. Principe de la prédistorsion

Cette méthode a été mise au point pour des modulations mono-porteuses, mais peut également être appliquée à l'OFDM. Cependant l'amplitude de saturation de l'amplificateur demeure une limite infranchissable, et un émetteur radio muni de cet amplificateur linéarisé aura tout de même des saturations et donc des erreurs dues à des non linéarités. Plusieurs moyens de réaliser cette prédistorsion ont été étudiés, dont un modèle polynomial [BENE96] et un modèle qui sépare la distorsion en deux, une sur le module du signal et une autre sur sa phase [ANDR01].

4.5. Correction à la réception

Une dernière possibilité est de corriger les distorsions introduites par les non linéarités dans le récepteur. L'avantage de cette méthode est qu'elle peut être utilisée sur des systèmes existants car il n'est pas nécessaire de modifier le codage ou les composants de l'émetteur. Seul le récepteur doit être modifié pour faire fonctionner un algorithme supplémentaire au niveau de l'égalisation. Et aucune modification des normes et protocoles existants est nécessaire.

Un algorithme itératif est proposé dans [YAMA98], et est adapté au cas de l'OFDM en couches (layered OFDM). La modulation en couches consiste à séparer les porteuses en plusieurs catégories, dont certaines sont plus protégées contre les erreurs que les autres. Par exemple dans une application de télévision numérique, deux couches peuvent être présentes : une couche bas débit avec un fort pouvoir de correction, qui fournit une image de qualité moyenne, et une couche haut débit avec moins de correction, qui permet aux récepteurs recevant un signal de bonne qualité de fournir une meilleure image. L'algorithme présenté s'appuie sur les porteuses à fort pouvoir de correction pour corriger de manière itérative les distorsions non linéaires sur les porteuses à faible pouvoir de correction.

Dans [NISH96], les auteurs proposent une autre méthode appelée postdistorsion. Celle-ci peut s'appliquer à n'importe quel système OFDM, et consiste en un module placé dans le récepteur après l'égalisation de canal. Ce module simule la chaîne OFDM complète, teste la transmission de différents symboles OFDM, et compare le résultat avec le symbole reçu. Ainsi on peut déterminer le symbole qui a été émis avec la plus grande probabilité.

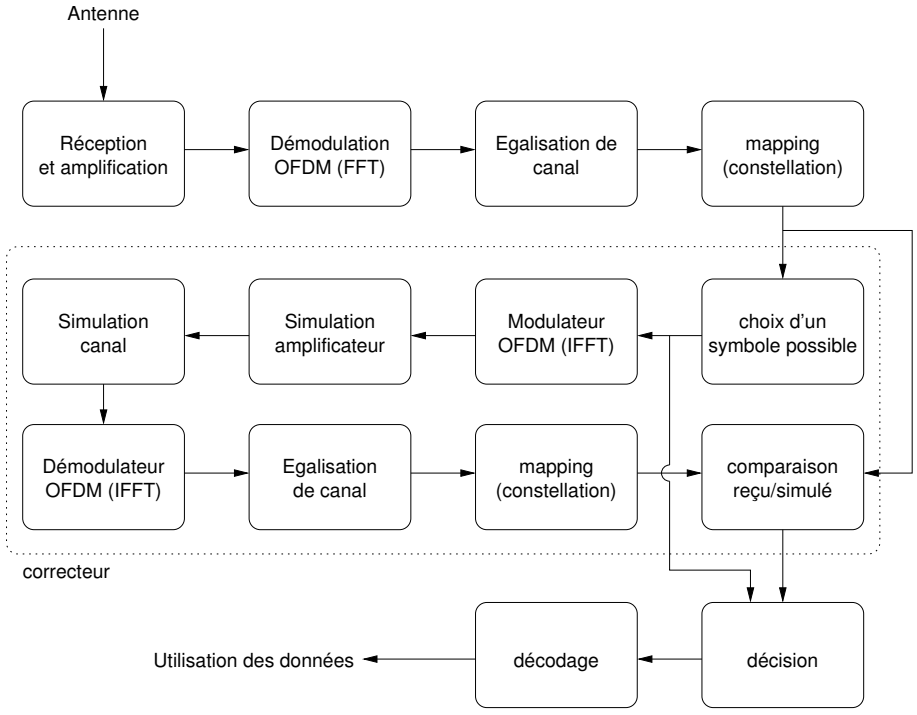

Figure 1.32. Principe d'une postdistorsion OFDM

Il n'est pas possible de simuler tous les symboles OFDM possibles car cela demanderait trop de calcul, donc des algorithmes sont chargés de limiter le nombre de tests à réaliser en choisissant soigneusement les candidats potentiels. Ces algorithmes sont améliorés dans [ATAR98] et permettent de réduire le nombre de tests, et donc de calculs, avec une augmentation négligeable

de l'erreur. Par contre dans les deux cas, la postdistorsion suppose que l'on connaît parfaitement le modèle de l'amplificateur et du canal afin de les simuler correctement.

Conclusion

Ce chapitre a tout d'abord présenté les modulations numériques, puis le principe et les avantages des modulations multiporteuses. Dans le cas de canaux sélectifs en fréquence, tels que les canaux radio en milieu urbain ou confiné ainsi que les liaisons filaires à haut débit, les modulations multiporteuses permettent de réduire les interférences entre symboles dûes aux trajets multiples des signaux. Cependant le facteur de crête élevé du signal temporel à transmettre le rend sensible aux non-linéarités de l'amplificateur, et les perturbations engendrées peuvent faire apparaître des erreurs de transmission.

Dans la dernière partie les méthodes actuellement proposées pour s'affranchir de ces problèmes sont exposées. La première technique repose sur une modification du signal temporel avant émission. Le but est de réduire les intermodulations générées par l'amplificateur, mais les erreurs de transmissions ne sont pas compensées. Un second ensemble de méthodes consiste à modifier le codage canal afin de générer un signal temporel avec un faible facteur de crête. Généralement ces techniques ont un faible rendement, surtout en augmentant le nombre de porteuses, et le débit utile de la transmission chute. D'autres méthodes modifient les symboles OFDM dans le domaine fréquentiel afin également de réduire le facteur de crête, également au détriment du débit utile. La prédistorstion est une autre méthode, efficace sur une modulation monoporteuse, mais qui a une efficacité bien moindre en multiporteuses en raison du facteur de crête élevé du signal. Enfin un dernier ensemble de méthodes effectue une correction à la réception pour retrouver le symbole émis. Ces méthodes ont l'avantage de ne pas modifier l'émetteur et le protocole de communication, mais augmentent dans de grandes proportions la complexité du récepteur.

C'est dans ce dernier contexte que nous allons réaliser un correcteur basé sur un réseau de neurones. Le chapitre suivant est donc consacré aux techniques neuronales, et surtout leur utilisation pour l'approximation de fonction.

Chapitre 2. Réseaux de neurones et approximation de fonction

Introduction

Ce chapitre introduit à l'approximation de fonctions multidimensionnelles à l'aide de réseaux de neurones. Tout d'abord la problématique est posée, puis certaines méthodes traditionnelles d'approximation sont brièvement présentées. Ensuite on aborde le principe des réseaux de neurones, ainsi que les différentes architectures qui ont été développées et leurs domaines d'application.

Le lecteur connaissant les techniques neuronales peut donc sauter ce chapitre, sauf éventuellement la section 4 p. 58 consacrée aux réseaux d'ordre supérieur qui sont moins connus.

Les réseaux de neurones utilisés dans le cadre de cette thèse ont eu, comme leur nom l'indique, pour source d'inspiration les neurones biologiques. Toutefois ils font maintenant partie de la panoplie des outils dont on dispose pour traiter des données et le mot "neurone" est parfois utilisé de manière abusive. On peut retenir de ces outils deux caractéristiques essentielles. Tout d'abord la tâche à effectuer par le réseau est décomposée en tâches élémentaires, réalisées par des neurones. Chaque neurone possède des entrées et une sortie. Les neurones peuvent être organisés en couches et être reliés entre eux.

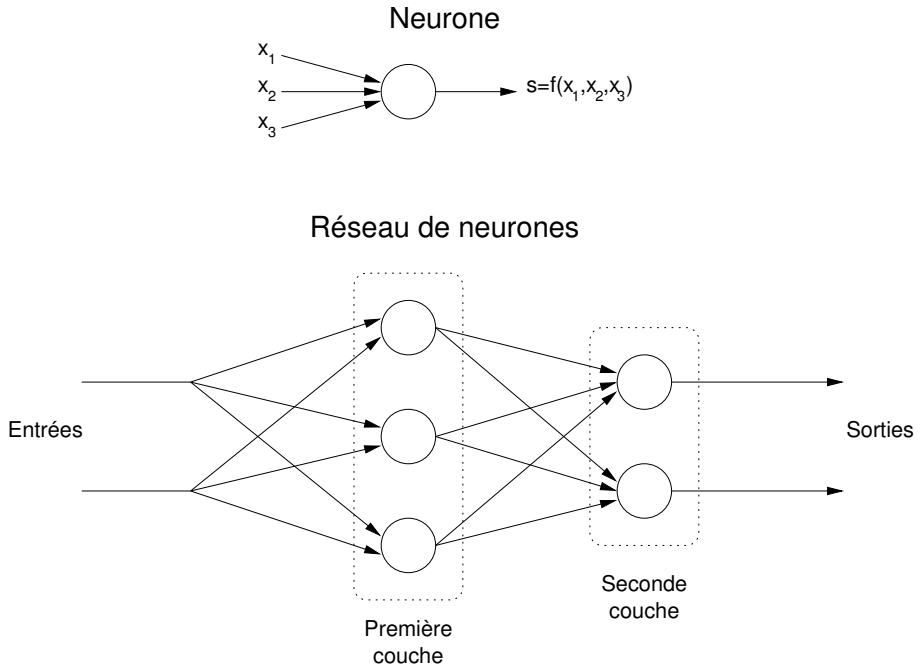

Figure 2.1. Schéma d'un neurone et d'un exemple de réseau

La seconde caractéristique d'un réseau de neurones est qu'il est adaptatif. Dans chaque neurone des paramètres peuvent être modifiées, et servent à adapter le réseau à une tâche particulière. Ces modifications sont fait lors d'une phase appelée apprentissage du réseau. Les

réseaux de neurones sont utilisés dans des domaines très variés tels que la reconnaissance d'écriture manuscrite [MOZA98], le traitement d'images [BURE91], et les télécommunications [MAN94][BALA95].

Dans ce chapitre l'accent est mis sur les aspects pratiques et la mise en oeuvre de telles méthodes, cependant des références vers des articles présentant les aspects plus théoriques et les démonstrations de validité des algorithmes seront données pour le lecteur intéressé.

1. Problème de l'approximation de fonction

Cette présentation générale s'appuie principalement sur les articles [BOSM96] et [POGG89].

1.1. Principe

Supposons que l'on veuille étudier un phénomène physique donné. L'état de ce phénomène peut être représenté par une série de grandeurs, que l'on peut regrouper dans un vecteur appelé vecteur de sortie, et noté $\mathbf{y}$. Cet état dépend de plusieurs paramètres extérieurs, que l'on peut regrouper dans un autre vecteur, appelé vecteur d'entrée, et noté $\mathbf{x}$. On supposera que le phénomène n'a pas de mémoire, c'est à dire que sa sortie $\mathbf{y}$ à un instant donné ne dépend que de son entrée $\mathbf{x}$ à ce même instant et non pas des entrées précédentes. Dans ce cas on peut représenter le phénomène physique comme une fonction ($\mathbf{y} = \mathbf{f}(\mathbf{x})$) et sous forme d'un schéma-bloc :

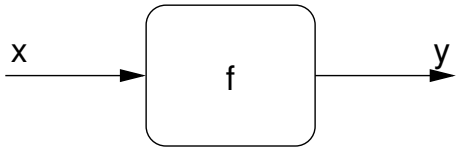

Figure 2.2. Schéma bloc représentant le phénomène physique à approximer

Si l'on ne sait pas modéliser précisément le phénomène physique mais que l'on aimerait disposer d'une simulation, on peut recourir à l'approximation de fonction. Le but est de créer une nouvelle fonction, $\mathbf{g}(\mathbf{x})$, que l'on connaît parfaitement et qui représentera au mieux possible la fonction $\mathbf{f}(\mathbf{x})$. On définit une mesure de l'écart entre les deux fonctions, que l'on appelle performance[5] . La performance la plus utilisée est l'erreur quadratique :

$$e_q = < \|\mathbf{f}(\mathbf{x}) - \mathbf{g}(\mathbf{x})\|^2 >_{|\mathbf{x}} \tag{2.1}$$

N'ayant pas accès à la fonction $\mathbf{f}(\mathbf{x})$, cette performance ne pourra pas être calculée. Par contre on dispose d'un jeu de mesures du phénomène physique que l'on peut représenter par une série de couples $(\mathbf{x_k}, \mathbf{y_k} = \mathbf{f}(\mathbf{x_k}))$. Dans ce cas il est possible de calculer une performance approchée, avec ces couples de mesures. Pour l'erreur quadratique, il s'agit de l'erreur quadratique moyenne :

[5]dans la littérature, le critère qu'on appelle ici "performance" est de temps en temps appelé "erreur". Dans ce mémoire, on appellera "erreur" la différence entre la sortie fournie par le réseau et la sortie voulue pour un exemple particulier, et "performance" le critère qui représente la qualité de l'approximation, basé sur l'erreur. L'erreur quadratique moyenne est une performance que l'on calcule en réalisant la moyenne des carrés des erreurs sur toute la base.

$$\epsilon = \frac{1}{n} \sum_{k=1}^{n} \|\mathbf{y_k} - \mathbf{g}(\mathbf{x_k})\|^2 \tag{2.2}$$

L'approximation de fonction sera faite à l'aide de ce jeu de mesures, et l'on cherchera la fonction $\mathbf{g}(\mathbf{x})$ qui minimise la fonction de performance ϵ.

1.2. Conditions

La fonction $\mathbf{f}(\mathbf{x})$ doit vérifier certaines propriétés pour être approximée. Un exemple intéressant est donné dans [BOSM96] : supposons que $\mathbf{f}(\mathbf{x})$ soit le répertoire téléphonique, qui associe un numéro de téléphone $\mathbf{y}$ au nom du propriétaire $\mathbf{x}$. On peut disposer du répertoire téléphonique d'une ville, qui correspond à un jeu de mesures $(\mathbf{x_k}, \mathbf{y_k})$. Même si l'on trouve une fonction $\mathbf{g}(\mathbf{x})$ qui approxime parfaitement $\mathbf{f}(\mathbf{x})$, c'est à dire qui a une erreur quadratique moyenne nulle sur le jeu de mesures, l'approximation ne pourra pas donner le numéro de téléphone d'un nouvel arrivant en ville. Ceci est tout simplement dû au fait que la fonction $\mathbf{f}(\mathbf{x})$ n'est pas continue, et que le numéro de téléphone correspondant à un nom donné ne donne aucune information sur les numéros des noms voisins.

Plus généralement, un algorithme d'approximation a besoin que pour deux entrées voisines $\mathbf{x_1}$ et $\mathbf{x_2}$ les sorties correspondantes $\mathbf{y_1}$ et $\mathbf{y_2}$ soient également voisines. Les fonctions respectant cette propriété sont appelées fonctions douces ("smooth functions" en Anglais). "Fonction douce" est une notion relative, mais une définition simple que l'on peut retenir et qui suffit généralement est une fonction dérivable par morceaux et de dérivée bornée. Fort heureusement la très grande majorité des fonctions représentant des phénomènes physiques respecte cette propriété en pratique.

Une seconde condition porte sur le choix du jeu de données qui servira à réaliser l'approximation. Celui-ci doit être représentatif des évolutions de la fonction $\mathbf{f}(\mathbf{x})$. Si l'on choisit le cas simple où l'on veut approximer une fonction à une dimension, sinusoïdale, et si les exemples choisis sont espacés de 2π, on remarque sur la figure 2.3 qu'une droite passant par les exemples peut être considérée comme une bonne approximation par l'algorithme :

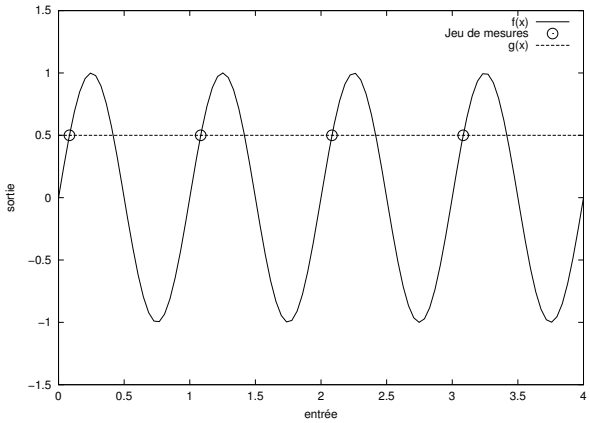

Figure 2.3. Approximation d'une sinusoïde avec un mauvais jeu de données

En effet avec ce jeu de données, l'erreur quadratique moyenne est nulle, alors que la fonction $\mathbf{g}(\mathbf{x})$ n'est pas une bonne approximation de la sinusoïde dans l'absolu. Pour détecter

ces anomalies, il est fréquent de définir deux jeux de données. Le premier, appelé base d'apprentissage, contient les données qui serviront à déterminer la fonction $\mathbf{g(x)}$. Le second, appelé base de validation, est distinct du premier, et servira simplement à vérifier que l'approximation se déroule correctement. Deux performances sont calculées, ϵ_a avec la base d'apprentissage et ϵ_v avec la base de validation. On cherche à minimiser ϵ_a et on surveille l'évolution de ϵ_v. Si un écart important est constaté entre les deux performances, ceci veut dire que l'approximation n'est pas bonne, et une cause possible est un mauvais choix de la base d'apprentissage.

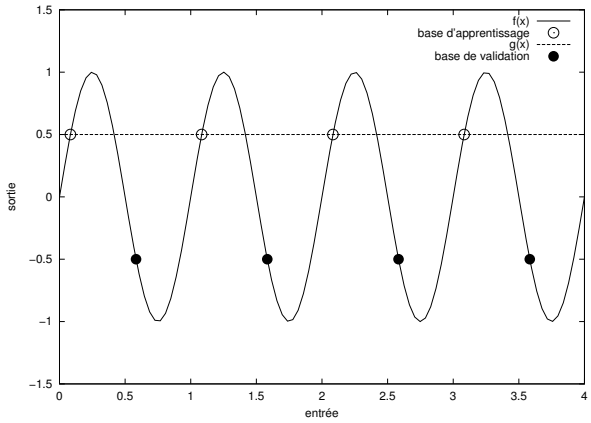

Figure 2.4. Détection d'une mauvaise approximation

Dans l'exemple ci-dessus, avec une base de validation décalée de π par rapport à la base d'apprentissage, on remarque un ϵ_a nul, avec un ϵ_v élevé, ce qui est révélateur d'une base non adaptée au problème. Malheureusement il n'existe pas de critère général permettant de dire si la base d'apprentissage est bien choisie. Il est nécessaire de connaître un peu le système que l'on veut approximer, et faire quelques tests pour déterminer si le jeu de données est pertinent ou non. Dans la plupart des cas, une base constituée de vecteurs sélectionnés aléatoirement donne de bons résultats, mais le choix de la taille de la base est toujours délicat, et dépend autant de la fonction à approximer que de la méthode utilisée pour l'approximation. Par exemple pour la sinusoïde une telle base pourrait convenir :

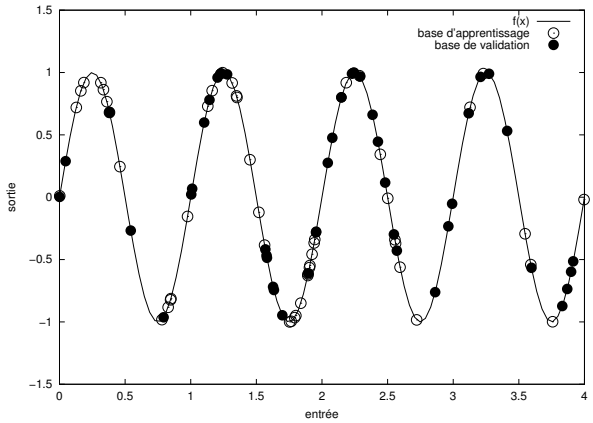

Figure 2.5. Bases possibles d'apprentissage et de validation pour une sinusoïde

1.3. Mise en oeuvre

Il existe de nombreuses méthodes d'approximation déterminant une fonction $\mathbf{g}(\mathbf{x})$ qui approxime dans certaines conditions la fonction $\mathbf{f}(\mathbf{x})$. En pratique il n'est pas possible d'avoir un système générique, capable de créer une fonction $\mathbf{g}(\mathbf{x})$ de n'importe quelle forme, et donc ces méthodes se limitent à une famille de fonctions $\mathbf{g}(\mathbf{x})$ possibles. Par exemple on peut envisager d'utiliser la famille des fonctions polynomiales de degré 3. On peut représenter cette famille par $\mathbf{g}(\mathbf{p}, \mathbf{x})$ où $\mathbf{x}$ est toujours l'entrée de la fonction, et $\mathbf{p}$ est un vecteur paramètre, qui caractérise la fonction choisie dans la famille. Dans l'exemple de la famille des fonctions polynomiales de degré 3, le vecteur $\mathbf{p}$ sera l'ensemble des coefficients du polynôme.

Finalement, une fois la famille choisie, le problème de l'approximation devient un problème d'optimisation numérique. En effet il se réduit à trouver le vecteur de paramètres $\mathbf{p}$ qui minimise :

$$\epsilon_a(\mathbf{p}) = \frac{1}{N_a} \sum_{k=1}^{N_a} \|\mathbf{g}(\mathbf{p}, \mathbf{x_k}) - \mathbf{y_k}\|^2 \tag{2.3}$$

où N_a est le nombre d'éléments dans la base d'apprentissage $(\mathbf{x_k}, \mathbf{y_k})$. Le nombre de paramètres (la dimension du vecteur $\mathbf{p}$) est le nombre de degrés de liberté de l'algorithme qui va déterminer la meilleure approximation. Plus ce nombre est grand, plus il sera possible de réaliser des fonctions différentes, et donc plus il sera possible d'approcher la fonction $\mathbf{f}(\mathbf{x})$. Par contre avec un nombre de degrés de liberté plus faible, la recherche de l'approximation demande moins de calcul.

De plus si le nombre de degrés de liberté est trop élevé, un autre problème survient, celui du surapprentissage, ou apprentissage par coeur. La fonction $\mathbf{g}(\mathbf{x})$ obtenue est plus complexe que la fonction d'origine, et est parfaitement égale à celle-ci aux points définis par les exemples fournis. Mais $g(x)$ ne généralise pas correctement, c'est à dire que les deux fonctions diffèrent beaucoup entre les points définis par la base d'apprentissage. Un exemple de ce phénomène de surapprentissage est illustré par la figure 2.6 :

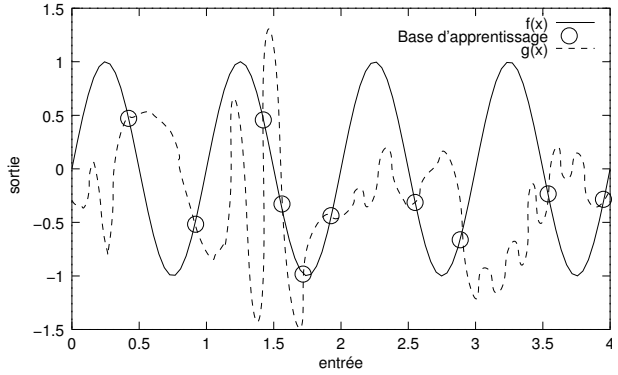

Figure 2.6. Surapprentissage avec une fonction sinusoïdale

Dans ce cas également le surapprentissage peut être détecté en définissant une seconde base, la base de validation, et en étudiant l'évolution de la performance sur cette seconde base. Pour éviter ce problème il faut limiter le nombre de degrés de liberté (tout en garantissant que les

fonctions $\mathbf{g}(\mathbf{x})$ pourront être assez complexes pour approximer correctement) et augmenter la taille de la base d'apprentissage.

Avant de présenter les techniques neuronales pour l'approximation de fonction, qui sont l'objet principal de ce chapitre, nous présenterons brièvement quelques méthodes classiques d'approximation.

1.4. Quelques méthodes d'approximation de fonction

Le but n'est pas d'énumérer toutes les méthodes d'approximation de fonctions, mais simplement d'en citer quelques unes, utilisant une approche paramétrique. Leur point commun est de dépendre d'un vecteur de paramètres, et la méthode consiste à trouver le vecteur paramètre permettant de trouver la meilleure approximation possible.

Une première méthode est celle des moindres carrés. On choisit une famille simple de fonctions qui dépendent du vecteur paramètre $\mathbf{p}$, et on détermine le minimum de la fonction d'erreur quadratique moyenne :

$$\epsilon_a(\mathbf{p}) = \frac{1}{N_a} \sum_{k=1}^{N_a} \|\mathbf{g}(\mathbf{p}, \mathbf{x_k}) - \mathbf{y_k}\|^2 \tag{2.4}$$

Ce minimum est déterminé en cherchant l'extremum de cette fonction, c'est-à-dire le vecteur $\mathbf{p}$ pour lequel toutes les dérivées partielles s'annulent :

$$\frac{\partial \epsilon_a}{\partial p_j} = 0 \; \forall j \in [1, N_p] \tag{2.5}$$

où p_j est la $j^{\text{ème}}$ composante de $\mathbf{p}$, et N_p est le nombre de paramètres (c'est à dire la dimension de $\mathbf{p}$). On doit donc résoudre le système d'équations suivant :

$$\sum_{k=1}^{N_a} (\mathbf{g}(\mathbf{p}, \mathbf{x_k}) - \mathbf{y_k}) \cdot \frac{\partial \mathbf{g}}{\partial p_j}(\mathbf{p}, \mathbf{x_k}) = 0 \; \forall j \in [1, N_p] \tag{2.6}$$

L'application principale de cette méthode est la régression linéaire. Dans ce cas l'expression des fonctions $\mathbf{g}(\mathbf{p}, \mathbf{x})$ est :

$$\mathbf{g}(\mathbf{p}, \mathbf{x}) = \sum_{l=1}^{N_e} p_l x_l + p_{N_e+1} \tag{2.7}$$

où N_e est la dimension de $\mathbf{x}$, et x_l est la $l^{\text{ème}}$ composante de $\mathbf{x}$. Ce sont en fait des fonctions affines. Le système d'équations à résoudre devient alors un système linéaire, d'inconnues p_j :

$$\begin{cases} \sum_{k=1}^{N_a}\{(\sum_{l=1}^{N_e} p_l x_{k,l}) + p_{N_e} - y_k\}x_{k,j} = 0 \quad \forall j \in [1, N_e] \\ \sum_{k=1}^{N_a}\{(\sum_{l=1}^{N_e} p_l x_{k,l}) + p_{N_e} - y_k\} = 0 \end{cases} \qquad (2.8)$$

où $x_{k,j}$ *est la* $j^{\text{ème}}$ *composante de* $\mathbf{x_k}$.

La méthode des moindres carrés est assez simple à mettre en place. Cependant elle n'est efficace qu'avec des familles de fonctions $\mathbf{g}(\mathbf{p}, \mathbf{x})$ simples, c'est à dire telles que la fonction de performance $\epsilon_a(\mathbf{p})$ ne présente qu'un seul extremum. Il est donc difficile d'appliquer cette méthode à des fonctions plus complexes, comme certains polynômes. De plus elle nécessite de connaître des informations a priori sur la forme de $\mathbf{f}(\mathbf{x})$ pour déterminer une famille de fonctions adaptée.

D'autres méthodes qui nécessitent moins de connaissances sur la fonction réelle réalisent des approximations locales. L'espace d'entrée est découpé en zones, et sur chaque zone la fonction est approximée par une fonction simple. L'utilisation la plus simple de cette méthode est l'approximation localement constante, dans laquelle chaque zone est approximée par une constante. En dimension 1, ceci donne des fonctions en escalier :

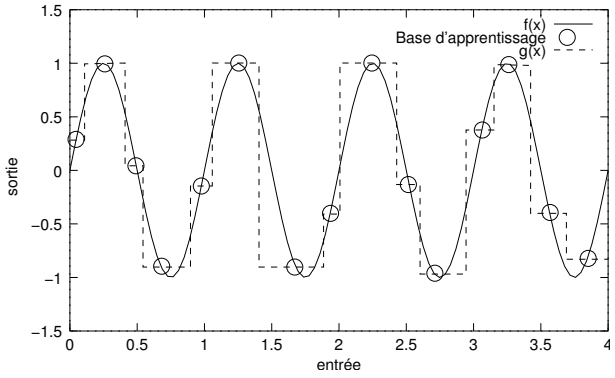

Figure 2.7. Approximation localement constante d'une sinusoïde

Pour avoir des modèles plus fins il est possible de choisir des fonctions légèrement plus complexes, comme les polynômes. Dans la méthode dite des splines [BOOR78], des polynômes de degré fixe (3 en général, mais des applications existent également avec des degrés 4,5 voire 6) permettent de réaliser des approximations plus fines et de garantir la continuité de l'approximation lors du passage d'une zone à l'autre. Ces méthodes sont très efficaces, mais le choix du nombre de zones et de leurs découpages dépend de l'application et peut s'avérer délicat.

Enfin la méthode "projection pursuit" [FRIE81] approxime la fonction $\mathbf{f}(\mathbf{x})$ par une somme de fonctions unidimensionnelles. Le vecteur d'entrée $\mathbf{x}$ subit une série de projections linéaires, chaque projection est fournie à une fonction, et les valeurs des fonctions sont sommées. L'expression de la fonction $\mathbf{g}(\mathbf{x})$ est la suivante :

$$\mathbf{g}(\mathbf{x}) = \sum_{i=1}^{N_f} \mathbf{b_i}(\sum_{k=1}^{N_e} x_k w_{i,k} + w_{i,N_e+1}) \tag{2.9}$$

où $\mathbf{b_i}$ sont des fonctions de base de l'approximation, N_f est le nombre de fonctions de base à utiliser, et $w_{i,k}$ sont les paramètres ajustables. N_f doit être précisé manuellement, et dépend de la complexité de la fonction à approximer. Ensuite les paramètres $w_{i,k}$ sont ajustés par un algorithme d'apprentissage. On peut montrer que cette méthode est capable d'approximer n'importe quelle fonction douce, pourvu que l'on choisit un N_f assez grand. En pratique on constate de bonnes performances même avec des N_f faibles, en particulier dans les cas où de la redondance est présente dans l'espace d'entrée, c'est-à-dire lorsque tous les vecteurs $\mathbf{x}$ utilisés n'occupent qu'un sous-espace de l'espace d'entrée (par exemple un plan dans un espace à 3 dimensions). Cependant les paramètres $w_{i,k}$ sont difficiles à optimiser et les algorithmes d'apprentissage demandent beaucoup de calcul, sans garantir de trouver la solution optimale.

2. Les réseaux GRBF

Les réseaux GRBF forment une première famille de réseaux de neurones. On va tout d'abord voir comment est constitué un tel réseau, puis ensuite leur apprentissage.

2.1. Architecture

Le réseau GRBF (Gaussian Radial Basis Functions, base de fonctions radiales gaussiennes)[6] est un réseau à deux couches [POGG89]. Les neurones de la première couche sont reliés aux entrées et ont chacun deux paramètres : un vecteur prototype $\mathbf{p}$ et un coefficient d'étalement σ strictement positif. La fonction réalisée, de forme gaussienne, est la suivante :

$$f_1(\mathbf{x}) = e^{-\frac{\|\mathbf{x}-\mathbf{p}\|^2}{2\sigma^2}} \tag{2.10}$$

Le vecteur prototype $\mathbf{p}$ définit un point dans l'espace d'entrée. La sortie s du neurone est égale à 1 pour une entrée $\mathbf{x}$ égale à $\mathbf{p}$, puis décroît vers 0 lorsque l'entrée s'éloigne de $\mathbf{p}$. La vitesse de décroissance est réglée par σ : plus le coefficient est petit et plus la fonction sera concentrée autour du point $\mathbf{p}$ et proche de 0 ailleurs.

[6] dans la littérature on peut trouver également le sigle RBF (Radial Basis Functions, base de fonction radiales) pour décrire ces réseaux, et le fait que les fonctions radiales soient gaussiennes est souvent implicite. Dans ce mémoire nous utiliserons le sigle GRBF.

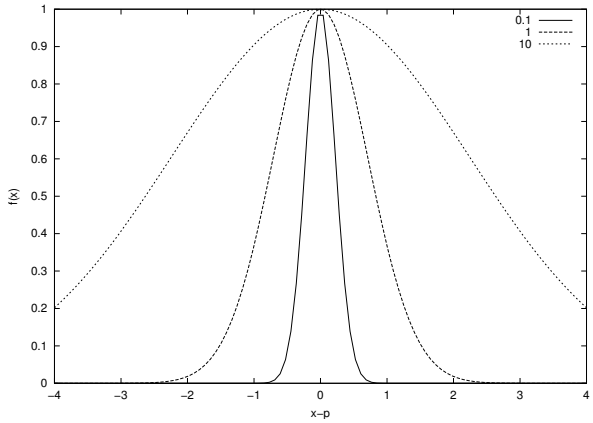

Figure 2.8. Gaussiennes obtenues avec différents étalements (σ= 0,1 1 et 10)

Les neurones de la seconde couche quand à eux calculent la sortie du réseau en effectuant une combinaison linéaire des sorties de ceux de la première couche, et un biais est ajouté au total. La fonction qu'ils réalisent est la suivante :

$$f_2(\mathbf{x}) = \mathbf{x} \cdot \mathbf{w} + b \tag{2.11}$$

où $\mathbf{x}$ est un vecteur composé des sorties de tous les neurones de la première couche, $\mathbf{w}$ est un vecteur de poids et b est le biais. $\mathbf{w}$ et b sont ajustés lors de l'apprentissage. Ainsi le réseau peut être représenté par le schéma suivant :

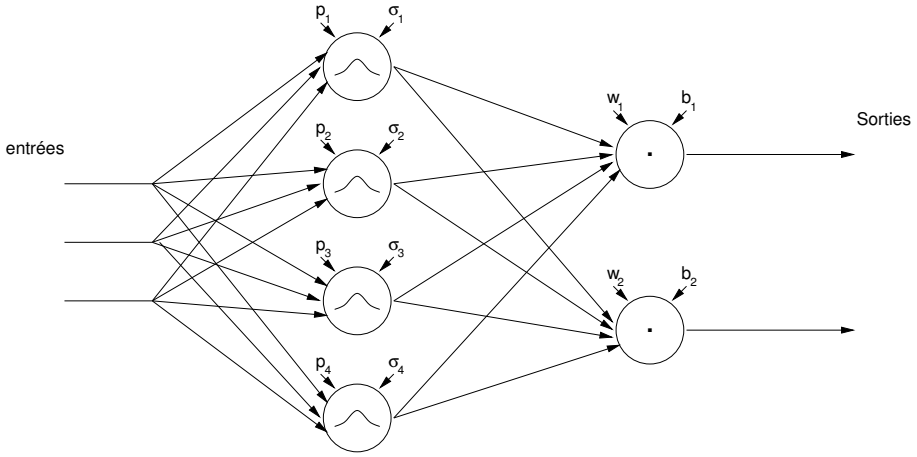

Figure 2.9. Schéma général d'un réseau GRBF

Et chaque sortie du réseau GRBF est donnée par la formule :

$$s_i(\mathbf{x}) = \sum_{j=1}^{N_g} w_{i,j} e^{-\frac{\|\mathbf{x}-\mathbf{p_j}\|^2}{2\sigma_j^2}} + b_i \tag{2.12}$$

où i est le numéro de la sortie (c'est à dire le numéro du neurone de la seconde couche dont on calcule la sortie), N_g le nombre de neurones de la première couche, $\mathbf{p_j}$ le vecteur prototype du

neurone numéro j de la première couche, σ_j son coefficient d'étalement, $w_{i,j},\ j = 1 \ldots N_g$ les N_g poids du neurone de sortie i, et b_i son biais.

Il a été montré que le réseau GRBF est un approximateur universel [PARK91], c'est à dire que le réseau est capable d'approximer n'importe quelle fonction douce avec une précision donnée, pourvu que l'on fournisse un nombre suffisant de neurones, et que l'on utilise un algorithme d'apprentissage adéquat. Lors de l'apprentissage d'un réseau GRBF deux problèmes se posent : la constitution de la première couche (choix du nombre de neurones, choix des prototypes et des coefficients d'étalement), et la détermination des poids et biais de la seconde couche.

2.2. Répartition des prototypes

Le choix du nombre de neurones sur la première couche et de leurs prototypes correspondants dépend fortement de la répartition des entrées, c'est à dire des vecteurs $\mathbf{x_k}$ de la base d'apprentissage. Pour établir ce choix, différentes méthodes sont applicables. On peut répartir les prototypes uniformément ou aléatoirement dans l'espace d'entrée, ou bien sélectionner aléatoirement quelques vecteurs de la base d'apprentissage. Ces méthodes sont très simples, mais ne donnent pas de bons résultats car généralement la répartition obtenue n'est pas adaptée à la base. On peut alors utiliser un algorithme de clustering statistique, tel que les k-means, afin de mieux les répartir dans l'espace d'entrée.

K-means est un algorithme itératif permettant de séparer une série de vecteurs dans différents groupes (les clusters) et chaque groupe est représenté par un prototype, placé en son centre. Dans l'exemple suivant la base est composée de 8000 vecteurs en deux dimensions, représentés ici sous la forme d'un nuage de points. 5 prototypes ont été déterminés par un algorithme k-means, indiqués dans la figure 2.10 par une croix.

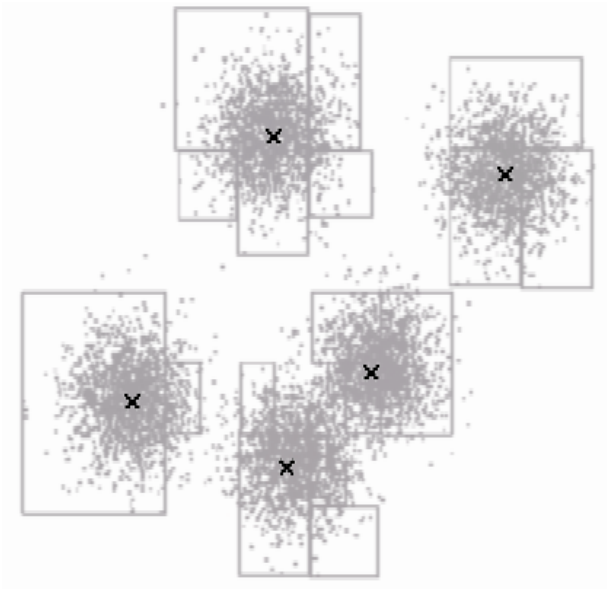

Figure 2.10. Exemple de clustering réalisé par un k-means

Le déroulement de l'algorithme k-means est le suivant : il faut tout d'abord choisir le nombre de prototypes désiré, empiriquement ou à l'aide de connaissances a priori sur la base d'apprentissage. Ensuite une série de prototypes initiaux est déterminée par l'une des méthodes précédentes (généralement un tirage aléatoire des exemples est utilisé). Puis chaque itération

est réalisée en deux étapes. Chaque vecteur d'entrée de la base est classé dans un cluster en déterminant le prototype le plus proche, puis après avoir parcouru toute la base, la position de chaque prototype est recalculée, en prenant le barycentre de tous les exemples situés dans son cluster. L'algorithme s'arrête lorsque le système est stable, c'est à dire quand aucun vecteur ne change de cluster d'une itération à la suivante. La solution trouvée par l'algorithme k-means est une solution locale, c'est à dire qu'il ne garantit pas de trouver une solution optimale. Cependant dans la plupart des cas pratiques la solution trouvée est satisfaisante, lorsque le nombre de prototypes décidés est adapté à la base.

D'autres algorithmes ont été étudiés et peuvent être également utilisés. Le problème à résoudre s'apparente à la construction d'un dictionnaire pour une quantification vectorielle. Une étude comparative des différentes solutions possibles est réalisée dans [FOUC02]. Elle s'intéresse particulièrement au cas des images, mais la plupart des méthodes peuvent s'appliquer à d'autres cas.

Le choix des coefficients d'étalement va influer sur la forme de la fonction réalisée. S'ils sont trop faibles, on verra apparaître des pics autour des points définis par les prototypes, et au contraire s'ils sont trop grands la fonction pourra difficilement prendre des valeurs différentes près de deux prototypes différents. Il est raisonnable de choisir une valeur qui soit de l'ordre de grandeur du carré de la distance entre deux prototypes voisins.

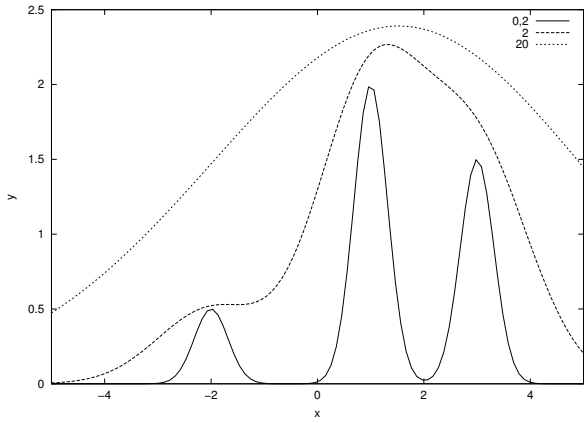

Figure 2.11. Exemples de fonctions réalisées par un GRBF avec différents coefficients d'étalement

2.3. Apprentissage de la seconde couche

Une fois les prototypes placés, il reste a déterminer les paramètres de la seconde couche, c'est-à-dire les poids $\mathbf{w_i}$ et biais b_i qui minimisent la fonction de performance du réseau. On peut calculer la sortie de la première couche pour chaque exemple de la base d'apprentissage :

$$z_{k,j} = e^{-\frac{\|\mathbf{x_k}-\mathbf{p_j}\|^2}{2\sigma_j^2}} \tag{2.13}$$

où k est le numéro de l'exemple et j celui de la composante de $\mathbf{z_k}$ calculée, c'est-à-dire le numéro du neurone de la première couche dont on calcule la sortie. Le critère à minimiser est l'erreur quadratique moyenne du réseau sur la base d'apprentissage et a donc pour expression :

$$\epsilon = \frac{1}{N_a}\sum_{k=1}^{N_a}\sum_{i=1}^{N_s}(s_i(\mathbf{x_k}) - y_{k,i})^2 = \frac{1}{N_a}\sum_{k=1}^{N_a}\sum_{i=1}^{N_s}(\mathbf{z_k}\cdot\mathbf{w_i} + b_i - y_{k,i})^2 \tag{2.14}$$

où $s_i(\mathbf{x_k})$ est la sortie du neurone i de la seconde couche ((2.12)), $y_{k,i}$ est la $i^{\text{ème}}$ composante de $\mathbf{y_k}$ (c'est à dire la sortie voulue pour l'exemple k sur le neurone i de la seconde couche) et N_s est le nombre de sorties. Un algorithme des moindres carrés peut être appliqué afin de trouver les paramètres $\mathbf{w_i}$ et b_i qui minimisent ce critère.

Cette méthode d'apprentissage qui consiste d'abord à sélectionner des prototypes puis ensuite à optimiser les poids de la seconde couche est simple à mettre en oeuvre. D'autres algorithmes, tels que EM (Expectation-Maximization) peuvent être appliqués aux GRBF afin de déterminer ces deux jeux paramètres en même temps, avec souvent une meilleure performance du réseau, mais aux pris d'un temps d'apprentissage plus long [BISH95]. Cependant toutes ces méthodes nécessitent de savoir par avance le nombre de prototypes à définir dans la première couche. D'autres méthodes ont donc été étudiées, de type incrémental, c'est à dire que des prototypes sont ajoutés dans la première couche au fur et à mesure des besoins de l'apprentissage.

2.4. Méthode incrémentale

Le principe d'une méthode incrémentale est de commencer avec un faible nombre de neurones sur la première couche (1 ou 2 par exemple) puis de commencer l'apprentissage, et ajouter au fur et à mesure des neurones sur la première couche jusqu'à ce que la fonction de performance du réseau soit en dessous d'un certain seuil, ou que le nombre de neurones ait atteint une valeur maximale. Généralement les réseaux ainsi obtenus possèdent moins de neurones et nécessitent donc moins de calcul lors de l'exploitation.

Plusieurs algorithmes ont été proposés dans la littérature. Dans [KURK95], les auteurs commencent avec un réseau ne comportant qu'un seul neurone sur la première couche. La détermination du prototype de ce neurone se fait de la manière suivante : chaque élément de la base est testé comme prototype, avec à chaque fois une détermination des poids correspondants dans la deuxième couche par une méthode des moindres carrés. C'est le prototype qui engendre l'erreur quadratique moyenne la plus faible qui est gardé. Ensuite cette erreur quadratique est comparée au seuil voulu. Si le seuil n'est pas atteint, un second neurone est ajouté par la même méthode, et ainsi de suite jusqu'à obtenir la performance voulue ou que le nombre de neurones a atteint un seuil maximal.

Avec cette méthode la première couche est figée, c'est à dire qu'une fois le prototype du nouveau neurone choisi il ne changera plus au cours du reste de l'apprentissage. Dans un autre algorithme proposé dans [FRIT94], les prototypes sont légèrement déplacés au cours de chaque itération, selon un algorithme de type "neural gas". Cet algorithme demande plus de calcul, mais permet d'affiner le modèle en prenant en compte l'influence des "anciens" neurones présents dans le réseau et ainsi de mieux répartir les prototypes.

2.5. Discussion

Les réseaux GRBF approximent localement les fonctions. En effet les neurones de la première couche ne produisent de valeurs significatives en sortie que dans une certaine zone de l'espace d'entrée (c'est à dire que leur sortie est proche de 1 pour une entrée proche de leur prototype

et proche de 0 ailleurs). La seconde couche permet d'associer une certaine sortie à chaque zone. Les réseaux GRBF sont donc particulièrement adaptés lorsque les vecteurs des entrées sont regroupés par zones, tel que dans l'exemple indiqué pour l'algorithme des k-means (figure 2.10). Dans cette situation les GRBF ont généralement de meilleures performances que les autres solutions neuronales, ou des performances équivalentes pour une complexité moindre. Par contre lorsque les vecteurs d'entrée sont répartis plus uniformément, ou lorsque le nombre de dimensions de l'espace d'entrée est important, le nombre de prototypes nécessaires augmente rapidement et les réseaux GRBF perdent leur intérêt. Enfin un autre avantage des réseaux GRBF est leur rapidité d'apprentissage, plus grande que pour le perceptron, que nous allons présenter maintenant.

3. Le perceptron multicouche

Le perceptron multicouche (PMC) est la deuxième grande famille de réseaux de neurones. Après avoir décrit l'architecture de ces réseaux on va aborder leur apprentissage, et le concept de rétropropagation de l'erreur.

3.1. Architecture

Un neurone de perceptron réalise un produit scalaire entre son vecteur d'entrées $\mathbf{x}$ et un vecteur de paramètres $\mathbf{w}$ appelé poids, y ajoute un biais b, et utilise une fonction d'activation f pour déterminer sa sortie [RUME86] :

$$y = f(\mathbf{x} \cdot \mathbf{w} + b) \tag{2.15}$$

Les fonctions d'activation doivent être de préférence strictement croissantes et bornées. Les fonctions classiquement utilisées sont la fonction linéaire, la tangente hyperbolique (f_1) et la fonction sigmoïde standard (f_2) :

$$\begin{aligned} f_1(x) &= tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \\ f_2(x) &= \frac{tanh(x) + 1}{2} \end{aligned} \tag{2.16}$$

La différence entre ces deux dernières fonctions est le domaine des valeurs prises, qui est de $]-1;1[$ pour la tangente hyperbolique et de $]0;1[$ pour la sigmoïde standard. La figure 2.12 montre la courbe de la tangente hyperbolique :

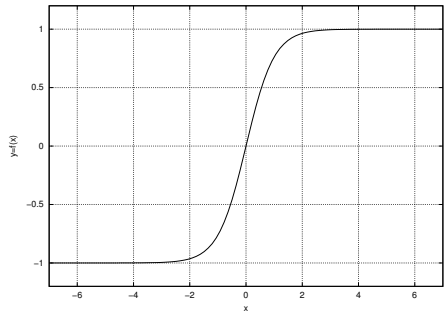

Figure 2.12. Tangente hyperbolique

Le perceptron est organisé en plusieurs couches. La première couche est reliée aux entrées, puis ensuite chaque couche est reliée à la couche précédente. C'est la dernière couche qui produit les sorties du PMC. Les sorties des autres couches ne sont pas visibles à l'extérieur du réseau, et elles sont appelées pour cette raison couches cachées.

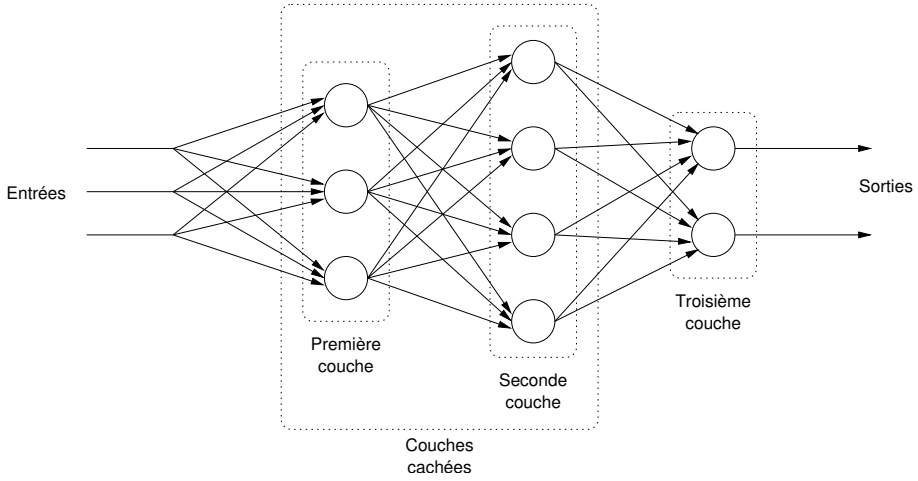

Figure 2.13. Schéma d'un PMC

Notons N_c le nombre de couches. L'indice l servira à désigner une couche, et notons n_l le nombre de neurones dans la couche l. L'indice i désigne un neurone. Le vecteur de poids du neurone i de la couche l est noté $\mathbf{w_{l,i}}$, et tous les vecteurs de poids d'une couche sont regroupés dans une matrice $\mathbf{W_l}$. En ce qui concerne les biais, on les considère souvent comme un poids supplémentaire associé à une entrée qui est toujours à 1. Ceci revient exactement au même, mais permet de simplifier les notations, et c'est donc ce que nous ferons dans le reste de ce mémoire. Notons $\mathbf{a_l}$ le vecteur regroupant les sorties des neurones de la couche l. Comme chaque couche est reliée à la précédente, $\mathbf{a_l}$ est également l'entrée de la couche $l+1$, en ajoutant l'entrée supplémentaire à 1 pour le biais. Par extension notons $\mathbf{a_0}$ l'entrée du réseau.

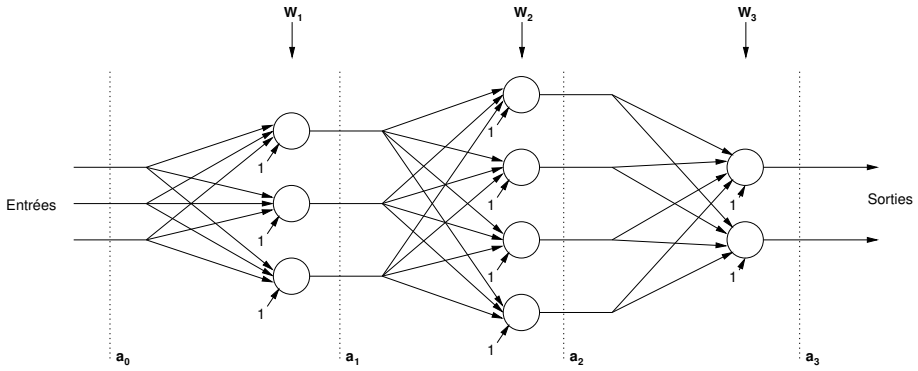

Figure 2.14. Notation des poids et des sorties des couches

Nous allons considérer de plus que tous les neurones d'une couche ont la même fonction d'activation, mais qu'elle peut différer d'une couche à l'autre. La fonction d'activation des neurones de la couche l est notée f_l et on définit une extension vectorielle de f_l par :

$$\mathbf{f_l}(\mathbf{a_l}) = (f_l(a_{l1}), f_l(a_{l2}), \ldots, f_l(a_{ln_l})) \tag{2.17}$$

On peut alors écrire la relation suivante pour exprimer la sortie d'une couche en fonction de son entrée :

$$\mathbf{a_l} = \mathbf{f_l}(\mathbf{a_{l-1}} \cdot \mathbf{W_l}) \tag{2.18}$$

Le calcul de la sortie du perceptron multicouche se fait de manière itérative. Il faut tout d'abord placer les entrées du réseau dans le vecteur $\mathbf{a_0}$, puis appliquer l'équation (2.18) avec $l = 1 \ldots N_c$ afin de calculer successivement $\mathbf{a_1}, \mathbf{a_2} \ldots, \mathbf{a_{N_c}}$. La sortie du réseau est alors $\mathbf{a_{N_c}}$.

Les fonctions qu'il est possible de réaliser avec un PMC sont diverses. Dans un perceptron à une couche, il n'y a pas de couche cachée, et l'unique couche relie les entrées du réseau aux sorties. Si la fonction d'activation utilisée est une sigmoïde, chaque sortie est une sigmoïde de produit scalaire. L'espace d'entrée est donc coupé en deux par un hyperplan. La sortie est égale à 1 d'un côté de l'hyperplan et à -1 de l'autre côté, lorsque l'on est situé à une certaine distance de celui-ci. Pour des points situés près de l'hyperplan, la transition est progressive. Dans l'exemple simple d'un réseau à deux entrées et une sortie composé d'un unique neurone, l'hyperplan est une droite. La figure 2.15 montre la sortie de ce neurone (axe vertical) en fonction de ses deux entrées (axes horizontaux) :

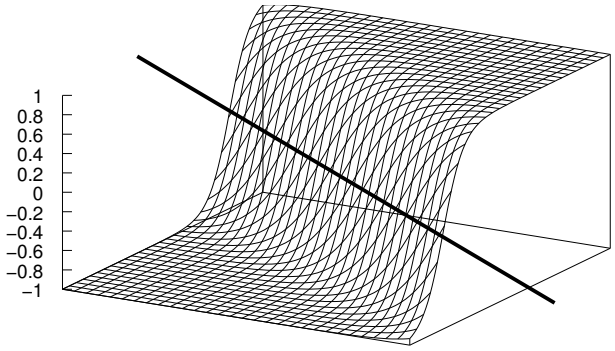

Figure 2.15. Sortie d'un perceptron à une couche en deux dimensions

Dans ce neurone, le biais fixe la distance entre l'origine et la droite (montrée figure 2.15), tandis que le vecteur poids (à deux dimensions) est orthogonal à la droite, et donc fixe sa direction, ainsi que la "largeur" de la zone de transition : plus le module du vecteur poids est élevé, plus la sortie évoluera rapidement de -1 à 1 en traversant la droite.

Dans un perceptron à deux couches, les sorties du réseau seront des combinaisons des sorties de la première couche, et on voit apparaître des intersections entre les zones définies par les neurones de la première couche.

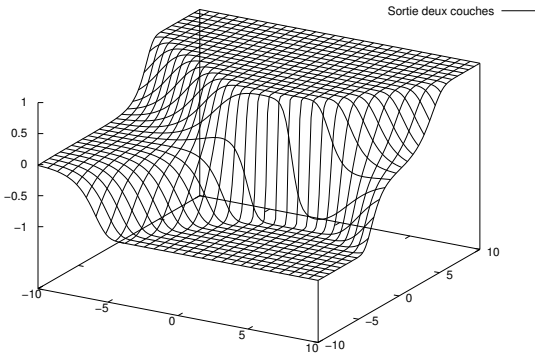

Figure 2.16. Sortie d'un perceptron à deux couches en deux dimensions

De manière plus analytique on peut considérer la sortie d'une couche (équation (2.18)) comme une fonction vectorielle, dont les éléments constituent une base de fonctions. En ajoutant une seconde couche, les sorties sont des combinaisons des différents éléments de cette base, qui peuvent éventuellement servir à nouveau de base pour une autre couche. Il y a une ressemblance entre une couche de PMC, qui permet une décomposition dans une base de fonctions tangentes hyperboliques et une couche de GRBF, qui elle permet une décomposition dans une base de fonctions gaussiennes.

Il a été montré qu'un perceptron à deux couches avec des fonctions d'activation intégrables au sens de Riemann non polynomiales sur la première couche et une fonction d'activation linéaire sur la seconde est un approximateur universel [HORN93]. Comme pour le GRBF, ceci veut dire que le réseau est capable d'approximer n'importe quelle fonction douce avec une précision donnée, pourvu que l'on fournisse un nombre suffisant de neurones dans la couche cachée. Cependant en pratique il n'est pas forcément possible d'approximer toute fonction, car dans certains cas le nombre de neurones nécessaire peut être très important, et le théorème dans l'article cité ne garantit pas que l'algorithme d'apprentissage pourra converger vers le résultat souhaité.

3.2. Rétropropagation

L'apprentissage d'un perceptron se fait avec une descente de gradient, algorithme décrit en annexe, section 1 p. 137. Dans le cas d'un perceptron à une couche l'expression de l'évolution des poids est assez simple. En effet l'erreur du réseau est de la forme :

$$e_{i,k} = f_1(\mathbf{w_{1,i}} \cdot \mathbf{x_k}) - y_{i,k} \tag{2.19}$$

où i est le numéro de la sortie et k celui de l'exemple de la base d'apprentissage. La performance du réseau, une erreur quadratique moyenne, est :

$$\epsilon = \frac{1}{N_a} \sum_{k=1}^{N_a} \sum_{i=1}^{N_s} e_{i,k}^2 \tag{2.20}$$

En appliquant à ces équations l'algorithme de descente du gradient ((A.1) et (A.2)), l'évolution des poids au cours d'une itération est donnée par [RUME86] :

$$\delta \mathbf{w_{1,i}} = -2\eta \sum_{k=1}^{N_a} e_{k,i} f_1'(\mathbf{w_{1,i}} \cdot \mathbf{x_k}) \mathbf{x_k} \tag{2.21}$$

Dans un perceptron multicouche il faut tenir compte de l'influence de plusieurs couches dans le calcul du gradient. Reprenons les notations utilisées dans la section 3.1 p. 53, en notant $\mathbf{a_{l,k}}$ la sortie de la couche l lorsque l'on applique en entrée l'exemple $\mathbf{x_k}$. En notant de plus $\mathbf{a_{0,k}} = \mathbf{x_k}$, on peut appliquer (2.18) pour calculer tous les $\mathbf{a_{l,k}}$ avec $l = 1 \dots N_c$ et $k = 1 \dots N_a$. La performance du perceptron est :

$$\epsilon = \frac{1}{N_a} \sum_{k=1}^{N_a} \|\mathbf{a_{N_c,k}} - \mathbf{y_k}\|^2 \tag{2.22}$$

En appliquant l'algorithme de descente du gradient ((A.1) et (A.2)) à cette performance et en utilisant (2.18), on montre qu'une méthode itérative permet de calculer facilement le vecteur gradient. En effet, on peut exprimer l'évolution des poids sous la forme [RUME86] :

$$\delta \mathbf{w_{l,i}} = -2\eta \sum_{k=1}^{N_a} e_{l,k,i} f_l'(\mathbf{w_{l,i}} \cdot \mathbf{a_{l-1,k}}) \mathbf{a_{l-1,k}} \tag{2.23}$$

où le terme d'erreur $\mathbf{e_{l,k}}$ de composantes $e_{l,k,i}$ est de la forme :

$$\begin{cases} \mathbf{e_{N_c,k}} = \mathbf{a_{N_c,k}} - \mathbf{y_k} \\ e_{l,k,i} = \sum_{j=1}^{n_{l+1}} e_{l+1,k,j} w_{l+1,j,i} f_{l+1}'(\mathbf{w_{l+1,j}} \cdot \mathbf{a_{l,k}}) \;\; l = 1 \dots N_c - 1, \quad i = 1 \dots n_l \end{cases} \tag{2.24}$$

On constate que l'évolution des poids est similaire à celle vue pour le perceptron à une couche, en définissant une erreur sur chaque couche du perceptron. L'erreur de la dernière couche est effectivement l'erreur du réseau, et pour chaque couche cachée les erreurs sont calculées à partir des erreurs de la couche suivante. Pour chaque neurone l'erreur est la somme des erreurs de chaque neurone de la couche suivante, pondérée par le poids qui le lie au neurone dont on calcule l'erreur et par la dérivée de la fonction d'activation. Pour cette raison cet algorithme est appelé rétropropagation de l'erreur.

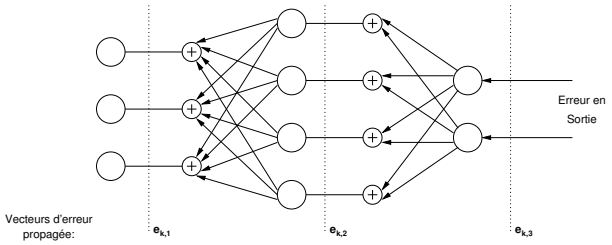

Figure 2.17. Principe de la rétropropagation

L'expression de la descente de gradient sous cette forme permet une réalisation simple de l'algorithme. A chaque itération, les différents vecteurs $\mathbf{a_{k,l}}$ puis la sortie du perceptron sont calculés en utilisant l'équation (2.18) en allant de la couche 1 à la couche N_c, puis les erreurs sont calculées en utilisant l'équation (2.24) en allant de la couche N_c à la couche 1. Enfin les poids de chaque neurone sont mis à jour en utilisant l'équation (2.23).

Lorsque l'on augmente le nombre de couches, on constate que l'algorithme d'apprentissage nécessite de plus en plus d'itérations pour converger vers un résultat. C'est pour cette raison que l'on dépasse rarement deux couches cachées dans un PMC. De plus une ou deux couches cachées suffisent généralement pour approximer ce que l'on veut. Si la descente de gradient est trop lente pour réaliser l'apprentissage, il est également possible d'utiliser un algorithme du second ordre, tel que celui de Levenberg Marquardt, présenté en annexe section 2 p. 139. Dans ce cas chaque itération demande plus de calculs, mais dans la plupart des cas le nombre d'itérations nécessaires pour converger est bien moindre.

3.3. Discussion

Un perceptron multicouche est capable d'approximer des fonctions de forme très différente. Contrairement au réseau GRBF l'approximation faite n'est pas locale, mais globale, et donc il sera plus adapté lorsque les vecteurs d'entrée ont une répartition assez uniforme dans l'espace d'entrée. Par contre si la fonction à approximer présente des variations assez localisées dans l'espace, un perceptron multicouche risque d'être plus complexe qu'un réseau GRBF avec les mêmes performances. Le choix du nombre de couches et du nombre de neurones est primordial dans un perceptron. En ajoutant des neurones ou des couches on améliore les capacités du réseau et donc la finesse de l'approximation, mais l'apprentissage devient plus long (particulièrement en augmentant le nombre de couches) et le risque de surapprentissage augmente. Généralement ces nombres sont déterminés expérimentalement, mais certains algorithmes itératifs existent également. On peut commencer avec un petit réseau et ajouter progressivement des neurones [BELI95], ou au contraire commencer avec un grand réseau et enlever des neurones [LUND97].

4. Réseaux d'ordre supérieur

4.1. Pourquoi utiliser l'ordre supérieur

Le but des réseaux d'ordre supérieur est d'améliorer les performances du perceptron sans recourir à un grand nombre de couches cachées ou de neurones. Les perceptrons monocouche sont en effets très limités dans leurs capacités d'apprentissage, et dans la plupart des applications une ou plusieurs couches cachées sont nécessaires pour obtenir une approximation avec la précision voulue. L'inconvénient est la plus grande complexité du réseau et surtout un temps d'apprentissage plus long. Une autre solution, présentée ici, est de créer un réseau de neurones capable de prendre en compte les corrélations d'ordre supérieur entre les entrées. Ou vu autrement il peut s'agir de combiner l'approche locale des GRBF et l'approche globale des PMC grâce à des noyaux polynomiaux. Les possibilités d'approximation du réseau de neurones sont plus importantes, tout en réduisant le nombre de neurones et de couches, et donc d'éviter de rendre l'apprentissage plus long. Plusieurs moyens existent et font l'objet de cette section.

4.2. Réseau SQUARE-MLP

Une première solution est de créer des entrées supplémentaires à un PMC, en calculant les carrés des entrées de départ. Cette architecture se nomme SQUARE-MLP (SQuare Unit Augmented, Radially Extended, MultiLayer Perceptron).

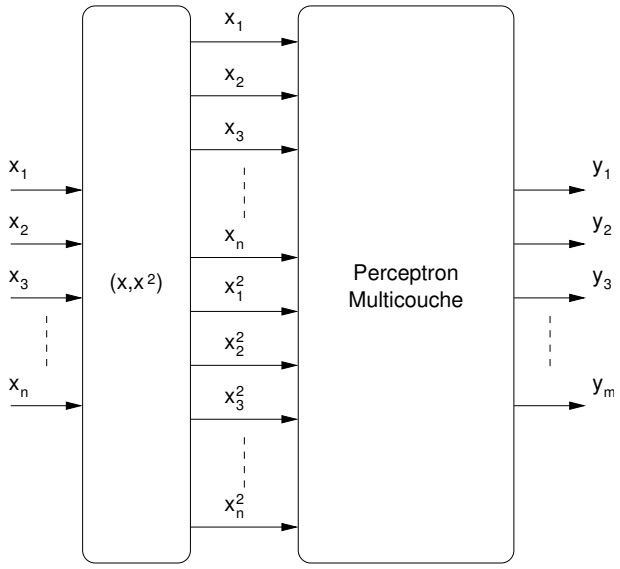

Figure 2.18. Principe du réseau Square-MLP

A partir du vecteur d'entrée $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ on construit un nouveau vecteur $(x_1, x_2, \ldots, x_n, x_1^2, x_2^2, \ldots, x_n^2)$. Ce vecteur est ensuite donné comme entrée à un PMC classique. Le vecteur donné en entrée du perceptron a une dimension deux fois plus grande que celui d'origine, et donc le perceptron possède deux fois plus de poids par neurone que s'il était utilisé seul. Cependant l'apport de ces entrées supplémentaires permet au réseau de réaliser une plus grande variété d'approximations avec un petit nombre de neurones. Considérons la fonction représentée figure 2.19 (comme précédemment, l'axe vertical est la sortie du réseau et les deux axes horizontaux sont ses deux entrées) :

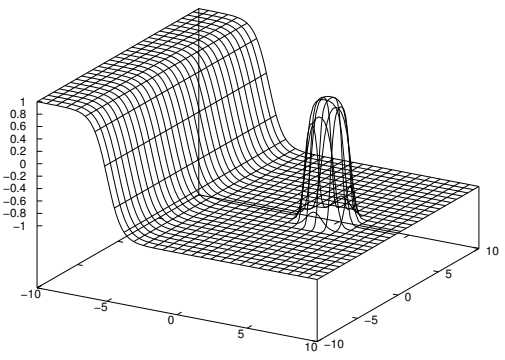

Figure 2.19. Fonction réalisée par un réseau Square-MLP simple (2 entrées, une couche cachée de 2 neurones, une sortie)

Cette fonction est complexe à réaliser aussi bien par un réseau GRBF qu'un perceptron multicouches. En effet le réseau GRBF sera capable d'approximer la partie de droite avec

un seul neurone, mais nécessitera un grand nombre de neurones pour approximer la partie gauche. Au contraire un perceptron multicouches pourra réaliser la partie gauche avec un seul neurone mais en nécessitera un grand nombre pour la partie droite. Avec un réseau Square-MLP, une couche cachée composée de deux neurones suivie d'une couche de sortie avec un neurone pourra approximer cette fonction avec de très bonnes performances.

Ce réseau a été testé dans un grand nombre de cas aussi bien théoriques que pratiques, et présente souvent de meilleures performances que les autres réseaux de neurones, avec une complexité moindre, en particulier dans les cas des fonctions présentant des variations à la fois sur de petites régions locales et sur des grandes divisions de l'espace d'entrée, comme dans le cas de la fonction ci-dessus [FLAK98].

Les réseaux Square-MLP peuvent être mis en oeuvre très rapidement et simplement, puisque les algorithmes déjà étudiés et implémentés pour les PMC peuvent être appliqués aux Square-MLP. La seule modification à faire est le calcul des entrées supplémentaires.

4.3. HPU

Le réseau HPU (Higher-order Processing Unit) utilise le même principe que le SQUARE-MLP, mais en calculant toutes les corrélations d'ordre supérieur entre les entrées, jusqu'à un certain ordre. Une sortie d'un réseau HPU est de la forme [GILE94] :

$$y = f(\omega_0 + \sum_j \omega_j x_j + \sum_{j,k\ j\leq k} \omega_{j,k} x_j x_k + \sum_{j,k,l\ j\leq k\leq l} \omega_{j,k,l} x_j x_k x_l + \cdots) \qquad (2.25)$$

où f est la fonction d'activation, linéaire ou de forme sigmoïde. Dans le cas d'un réseau d'ordre supérieur une fonction d'activation en sigmoïde ou tangente hyperbolique n'est pas obligatoire, car même avec une fonction f linéaire la fonction effectuée par le réseau est polynomiale et permet d'approximer de nombreuses fonctions. Ainsi son choix va dépendre de l'application. L'ordre du réseau est le degré du terme de plus haut degré. La réalisation d'un réseau HPU peut s'inspirer de celle d'un SQUARE-MLP :

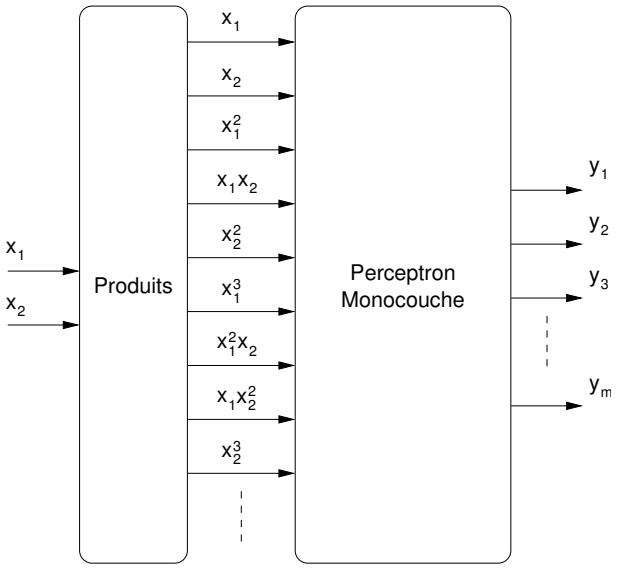

Figure 2.20. Mise en oeuvre d'un réseau HPU

En effet l'équation (2.25) correspond à la sortie d'un perceptron monocouche auquel on fournit en entrée tous les produits possibles entre les entrées jusqu'à l'ordre du réseau HPU. Comme pour le SQUARE-MLP, il est donc possible d'utiliser les algorithmes existants pour le perceptron, et d'ajouter simplement un module qui calcule ces produits avant de les fournir au réseau de neurones. Si la fonction f est linéaire les poids peuvent être déterminés plus simplement avec des moindres carrés.

Il est possible de créer un réseau multicouches dont chaque couche est constituée d'un HPU. Ceci dit en pratique cette approche rend le réseau compliqué et présente moins d'intérêt face à un PMC.

L'inconvénient du HPU réside dans le nombre de poids nécessaire. Celui-ci évolue de manière exponentielle avec le nombre d'entrées N_e et l'ordre du réseau HPU O_r. En effet, on peut montrer que ce nombre de poids est égal à [JOYD92] :

$$\sum_{i=0}^{O_r} \left(\begin{array}{c} N_e + i - 1 \\ i \end{array} \right) = \left(\begin{array}{c} N_e + O_r \\ O_r \end{array} \right) \tag{2.26}$$

où () est la notation combinatoire. Ce nombre de poids est donc égal au nombre de combinaisons de O_r éléments parmi $N_e + O_r$. Donc cette architecture ne peut être utilisée en pratique qu'avec un faible nombre d'entrées et pour des ordres peu élevés. Mais dans ces cas il présente quelques avantages, tels qu'un apprentissage rapide (ce réseau n'a pas de couche cachée) et une bonne adaptation aux problèmes présentant des invariances géométriques [GILE94].

4.4. Pi-Sigma

Le réseau Pi-Sigma est une réponse au problème du nombre important de poids du réseau HPU lorsque le nombre d'entrées ou l'ordre augmente. Un réseau Pi-Sigma ne comporte qu'une sortie scalaire, mais on peut créer un réseau avec N_s sorties en dupliquer le réseau N_s fois, chacun calculant une sortie. La sortie d'un réseau Pi-Sigma est donnée par [SHIN91] :

$$y = f(\mathbf{x}) = f(\prod_{i=1}^{O_r} (\mathbf{w_i} \cdot \mathbf{x} + b_i)) = f(\prod_{i=1}^{O_r} (\sum_{j=1}^{N_e} w_{i,j} x_j + b_i)) \tag{2.27}$$

où f est la fonction d'activation (linéaire ou non) et O_r est l'ordre du réseau. Comme pour le PMC (p. 54), le biais sera considéré comme un poids supplémentaire, lié à une entrée toujours à 1. La sortie est fonction d'un produit de sommes, d'où le nom Pi-Sigma. L'intérêt du réseau pi-sigma est le nombre réduit de poids par rapport à un HPU. En effet, celui-ci est égal à :

$$(N_e + 1) O_r \tag{2.28}$$

Un réseau pi-sigma est donc mieux adapté qu'un HPU pour des systèmes avec un grand nombre d'entrées, ou si l'on veut faire intervenir des corrélations d'ordre élevé entre les entrées.

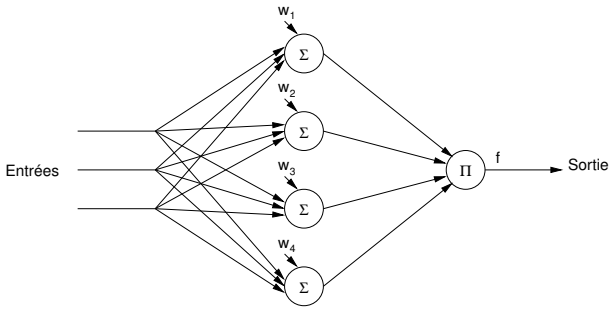

Figure 2.21. Architecture d'un réseau Pi-Sigma

Le réseau Pi-Sigma peut être décomposé en deux couches. La première couche est composée de O_r neurones, chaque neurone effectuant une somme pondérée des entrées et y ajoutant un biais. Dans la seconde couche un seul neurone est présent, et celui-ci calcule le produit des sorties des neurones de la première couche, y applique la fonction d'activation, et sa sortie est celle du réseau. Seuls les neurones de la première couche ont des poids ajustables.

L'apprentissage d'un réseau Pi-Sigma peut se faire avec un algorithme de descente de gradient, mais celui-ci nécessite quelques modifications, à cause du produit réalisé par le neurone de la seconde couche. Différents algorithmes sont étudiés en détail dans [JOYD92], et seul le principe de l'un d'entre eux sera évoqué ici. Nous appellerons $(\mathbf{x_k}, \mathbf{y_k})$ les éléments de la base d'apprentissage, e_k les erreurs correspondantes et ϵ la performance (une erreur quadratique moyenne). Comme pour le PMC nous considérons que le biais est un poids supplémentaire associé à une entrée à 1, et donc celui-ci n'apparaît plus dans les formules suivantes. Si l'on applique l'algorithme de descente du gradient à ce réseau (équations (A.1) et (A.2)) on obtient pour les mises à jour des poids :

$$\delta \mathbf{w_i} = -\eta \sum_{k=0}^{N_a} e_k d_f \prod_{j \neq i} (\mathbf{x_k} \cdot \mathbf{w_j}) \mathbf{x_k}$$

où (2.29)

$$d_f = f'(\prod_{j=1}^{O_r} (\mathbf{x_k} \cdot \mathbf{w_j}))$$

Si l'on applique directement l'algorithme de descente du gradient au réseau Pi-Sigma, c'est à dire si l'on ajoute les $\delta \mathbf{w_i}$ à tous les poids à chaque itération, on constate que l'algorithme ne converge que pour des faibles valeurs de η. Au delà l'algorithme est très instable. Donc deux autres approches sont proposées. Dans la première, dite aléatoire, à chaque itération un seul des O_r neurones, sélectionné aléatoirement, est mis à jour et les poids des autres neurones ne sont pas modifiés. Dans la seconde approche, dite asynchrone, la mise à jour des poids se fait successivement sur chaque neurone pour chaque exemple. C'est à dire que pour chaque exemple de la base d'apprentissage on calcule $\delta \mathbf{w_1}$, puis la nouvelle valeur de $\mathbf{w_1}$, puis $\delta \mathbf{w_2}$, puis la nouvelle valeur de $\mathbf{w_2}$ etc... L'itération se termine lorsque tous les exemples ont été parcourus. Ces deux approches permettent de converger plus rapidement qu'en appliquant strictement l'algorithme de descente du gradient avec un η faible.

Le réseau Pi-Sigma a été testé avec de nombreux exemples de bases d'apprentissage dans [JOYD92], et les simulations montrent qu'il est capable de réaliser un grand nombre de fonctions complexes avec un nombre de neurones et de poids réduit par rapport au HPU ou au perceptron multicouches.

4.5. RPN

Le réseau Pi-Sigma permet d'approximer un grand nombre de fonctions, en utilisant des corrélations d'ordre élevé entre les entrées avec un nombre réduit de poids. Cependant ce réseau n'est pas un approximateur universel, c'est à dire que certaines fonctions ne pourront pas être approximées par le réseau, quel que soit le nombre de neurones choisis. Le réseau RPN (Ridge Polynomial Network) [SHIN92] est un autre réseau basé sur le Pi-Sigma, qui lui est approximateur universel.

Le principe d'un réseau RPN est de calculer la somme des sorties de plusieurs réseaux Pi-Sigma. Chaque réseau Pi-Sigma ne possède pas de fonction d'activation (c'est à dire que $f(x) = x$ dans les formules précédentes) et la fonction d'activation est placée après la somme. Dans la figure 2.22 chaque PSN_i représente un réseau Pi-Sigma d'ordre i avec une fonction d'activation linéaire.

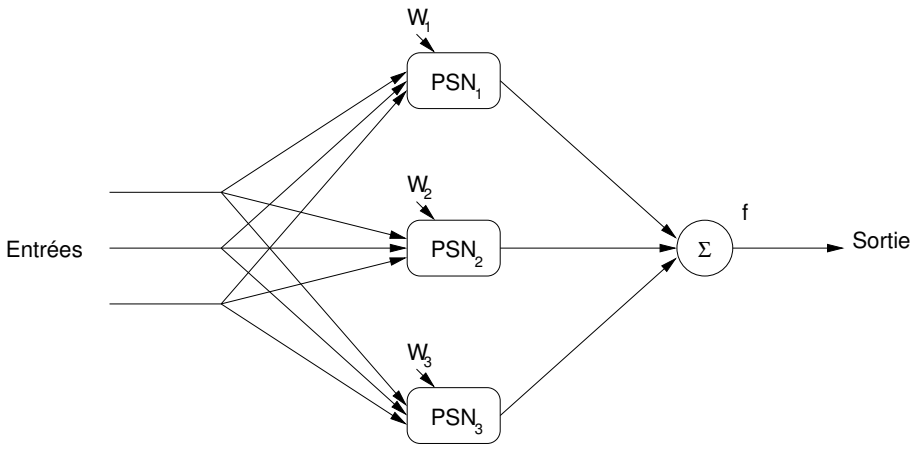

Figure 2.22. Architecture d'un réseau RPN

Un réseau RPN d'ordre O_r est composé de O_r réseaux Pi-Sigma, d'ordres respectifs $1, 2, 3, \ldots, O_r$. La sortie d'un réseau RPN est donnée par :

$$y = f(\sum_{i=1}^{O_r} p_i(\mathbf{x}))$$

où (2.30)

$$p_i(\mathbf{x}) = \prod_{j=1}^{i} (\mathbf{w_{i,j}} \cdot \mathbf{x} + b_{i,j})$$

L'apprentissage d'un réseau RPN se fait de manière itérative. Tout d'abord seul le premier réseau Pi-Sigma, d'ordre 1, est placé dans le réseau. Un apprentissage est réalisé sur ce réseau.

Ensuite les poids du premier réseau Pi-Sigma sont figés, et le second réseau, d'ordre 2, est intégré dans le réseau RPN. Un apprentissage est réalisé, puis ses poids sont figés, et ainsi de suite. L'apprentissage est stoppé lorsque la précision voulue est atteinte ou lorsque l'ordre du réseau RPN a atteint une certaine limite. L'algorithme d'apprentissage utilisé sur chaque réseau Pi-Sigma peut être l'un de ceux évoqués dans la section 4.4 p. 61. Chaque ajout de réseau Pi-Sigma permet d'affiner l'approximation effectuée par le réseau RPN.

Le réseau RPN permet de réaliser de meilleures approximations qu'un réseau Pi-Sigma, tout en gardant un nombre de poids inférieur à celui d'un réseau HPU d'ordre équivalent. Quelques exemples de problèmes où le réseau RPN a les meilleures performances sont donnés dans [SHIN95].

4.6. Discussion

Les réseaux d'ordre supérieur permettent d'approximer certaines fonctions dans de meilleures conditions (avec de meilleures performances, ou avec une complexité ou un temps d'apprentissage moins long) que les réseaux plus classiques, tels que les perceptrons et les GRBF. Les différentes architectures présentées ici ont chacune des avantages et des inconvénients aux niveaux des capacités d'approximation et de la complexité de calcul. Chacune est donc adaptée à des problèmes spécifiques. Malheureusement il n'existe pas de méthode pour déterminer par avance quelle architecture sera adaptée à un problème donné, et généralement il faut tester plusieurs architectures afin de choisir celle qui offre les meilleures performances.

Conclusion

Dans ce chapitre, le problème de l'approximation de fonction tel qu'il se posera dans cette thèse a été posé, et plusieurs méthodes basées sur les réseaux de neurones ont été présentées. Tout d'abord les architectures classiques de réseaux, les GRBF et les PMC ont été présentées, puis on s'est intéressé plus particulièrement aux réseaux d'ordre supérieur, qui peuvent être utiles dans de nombreux cas. On a ici insisté sur les aspects pratiques de mise en oeuvre de ces réseaux de neurones, tout en donnant des références vers des études plus théoriques.

Après la présentation du problème des non linéarités dans les multiporteuses dans le premier chapitre, et des réseaux de neurones pour l'approximation de fonctions dans le second chapitre, il est maintenant possible de voir comment nous proposons d'utiliser ces derniers pour résoudre le problème étudié, à savoir la compensation des effets de la non-linéarité de l'amplificateur dans une transmission OFDM.

Chapitre 3. Compensation des non-linéarités dans le domaine fréquentiel

Introduction

Dans ce chapitre une première méthode de compensation des non-linéarités en OFDM est proposée. Un correcteur basé sur un réseau de neurones est placé dans le récepteur, et celui-ci permet de corriger certaines erreurs causées par la non-linéarité de l'amplificateur, et ainsi améliorer la qualité de la transmission.

Tout d'abord le principe de la méthode est présenté, puis le protocole expérimental sera décrit. Ensuite nous verrons comment a été effectué le choix de l'architecture neuronale pour enfin présenter les résultats de cette méthode.

1. Principe de la méthode fréquentielle

1.1. Rôle et emplacement du réseau de neurones

En étudiant un système de communication numérique, on se rend compte qu'un système de compensation des non-linéarités peut être placé en divers endroits. Il est possible par exemple d'intervenir au niveau du codage source. Les réseaux de neurones ont déjà été étudiés pour le décodage source et il a été prouvé qu'ils peuvent accomplir cette tâche [CHAN98][JIAN95]. L'utilisation de réseaux de neurones pour créer ou utiliser un codage capable de corriger les erreurs particulières dues aux non-linéarités est un sujet de recherche intéressant mais ce n'est pas la voie qui a été choisie ici.

Il est également possible d'intervenir au niveau du canal. Dans les modulations monoporteuses, des réseaux de neurones ont été étudiés pour réaliser l'égalisation non linéaire de canal ainsi que la prédistorsion [LANG00]. La prédistorsion a déjà été évoquée dans le paragraphe 4.4 p. 37, et son utilisation en multiporteuses est limitée par la saturation de l'amplificateur, puisque la puissance de crête du signal à amplifier est souvent bien supérieure à la puissance de saturation. Par contre l'emploi d'un réseau de neurones pour réaliser une égalisation non linéaire du canal semble plus intéressante pour une application en multiporteuses.

Comme évoqué dans la section 2.3 p. 23, l'égalisation du canal en OFDM peut se réaliser très simplement, à l'aide d'une multiplication des symboles reçus. L'idée est donc de conserver cette simplicité, en séparant l'égalisation du canal et celle de la non-linéarité de l'amplificateur. La présence de blocs non linéaires dans la chaîne impose un ordre précis de ceux-ci. En effet les compensations doivent être placées dans l'ordre inverse de celui des effets compensés :

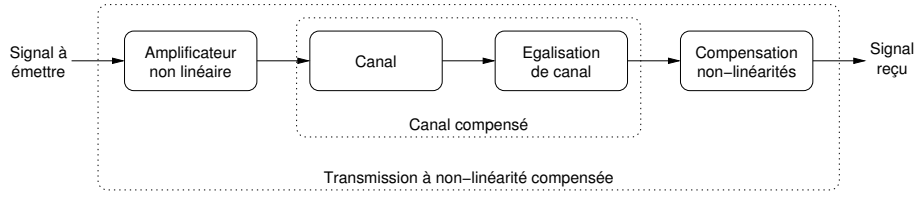

Figure 3.1. Ordre des différentes compensations dans une chaîne de transmission non linéaire

Ainsi le réseau de neurones sera inséré dans le récepteur OFDM après l'égalisation du canal et avant le seuil de décision et le décodage. La figure 3.2 présente le système plus en détail (TFD signifie Transformée de Fourier Discrète) :

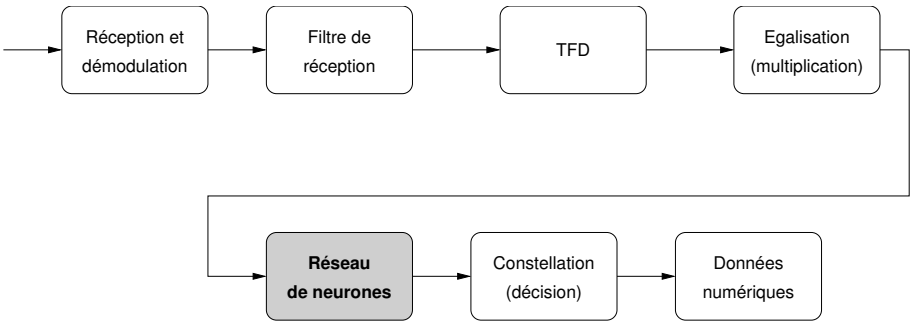

Figure 3.2. Schéma d'un récepteur OFDM avec réseau de neurones pour la compensation des non-linéarités

Le réseau de neurones travaille dans le domaine fréquentiel, et ne devra donc pas simplement réaliser une inversion de la fonction non linéaire temporelle de l'amplificateur : il devra prendre en compte les aspects fréquentiels de la non-linéarité.

1.2. Expression de la non-linéarité de l'amplificateur dans le domaine fréquentiel

Le but de cette section n'est pas de calculer précisément la non-linéarité dans le domaine fréquentiel, mais plutôt d'avoir une idée de son expression afin de déterminer comment le réseau de neurones devra être adapté.

Pour modéliser un amplificateur non linéaire, on peut utiliser les conversions AM/PM et AM/AM (AM : amplitude modulation, PM : phase modulation). Si l'on appelle $s(t)$ le signal à l'entrée de l'amplificateur et $s_0(t)$ celui à sa sortie, ces conversions sont définies par :

$$\begin{aligned} & s_0(t) = s(t) \cdot g(|s(t)|) \\ & \text{où } g(r) = \frac{a(r)e^{i\phi(r)}}{r} \end{aligned} \tag{3.1}$$

$a(r)$ est la conversion AM/AM, ou non-linéarité d'amplitude, et représente l'évolution du module de la sortie en fonction de celui de l'entrée, le cas linéaire correspondant à $a(r) = \nu r$ (où ν est le gain de l'amplificateur). $\phi(r)$ est la conversion AM/PM, ou non-linéarité de phase, et représente le déphasage en sortie en fonction du module de l'entrée, le cas linéaire correspondant à $\phi(r) = 0$. Un modèle utilisé pour les amplificateurs à semi-conducteurs est le SSPA (Solid State Power Amplifier) [RAPP91] :

$$\begin{cases} a(r) = \dfrac{\nu r}{(1 + (\frac{\nu r}{A_0})^{2p})^{\frac{1}{2p}}} \\ \phi(r) = 0 \end{cases} \tag{3.2}$$

p définit l'ordre de la non-linéarité, en général on choisit 1, 2 ou 3. ν est le gain dans le domaine linéaire, et A_0 est l'amplitude de saturation (en sortie). La non-linéarité de phase est nulle avec ce modèle. D'autres modèles représentant des amplificateurs présents dans les satellites (TWT pour Traveling-Wave Tube, ou TOP pour Tube à Ondes Progressives) ont une non-linéarité de phase non nulle.

Pour l'étude de la non-linéarité de l'amplificateur dans le domaine fréquentiel, nous supposerons que l'égalisation du canal est réalisée parfaitement, et donc allons retirer le canal de la modélisation. Ensuite nous cherchons à exprimer le symbole OFDM reçu $\mathbf{d}$ après égalisation du canal en fonction du symbole OFDM émis $\mathbf{c}$.

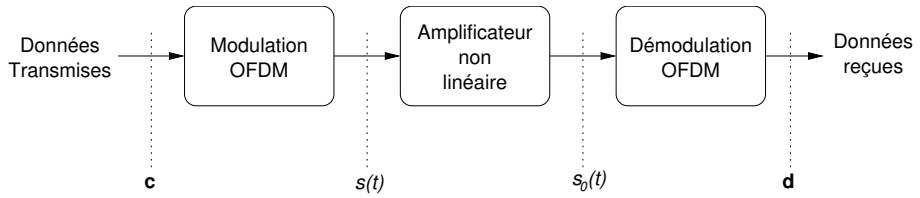

Figure 3.3. Modèle utilisé pour le calcul de l'expression de la non-linéarité dans le domaine fréquentiel

Les composantes du symbole OFDM émis sont $c_j,\ j = 0 \dots N_p - 1$, et celles du symbole reçu $d_k,\ k = 0 \dots N_p - 1$, N_p étant le nombre de porteuses. Le signal temporel $s(t)$ est la transformée de Fourier inverse du symbole OFDM (équations (1.19) et (1.20)) :

$$s(t) = \frac{1}{N_p} \sum_{j=0}^{N_p-1} c_j e^{2i\pi(f' + \frac{j}{T_S})t} \tag{3.3}$$

où T_S est la durée du symbole OFDM. Afin de simplifier l'écriture on ne calcule la valeur du signal que pour le premier symbole OFDM, et donc on ne s'intéresse qu'aux valeurs de $s(t)$ pour $t \in [0, T_S]$. De plus la fréquence de la première porteuse a été notée $f' = f_0 - N_p/2T_S$. Le modèle d'amplificateur non linéaire que l'on va utiliser est le modèle SSPA (équations (3.1) et (3.2)) :

$$\begin{aligned} & s_0(t) = a(|s(t)|)s(t) \\ & \text{où } a(|s(t)|) = \frac{\nu}{(1 + (\frac{\nu|s(t)|}{A_0})^{2p})^{\frac{1}{2p}}} \end{aligned} \tag{3.4}$$

La fonction a étant paire par construction[7] , son développement en série entière est de la forme :

$$a(|s(t)|) = \sum_{l=0}^{\infty} a_l |s(t)|^{2l} \tag{3.5}$$

[7]la fonction a n'est définie que sur $[0, +\infty[$ mais comme sa dérivée est nulle en 0 on peut la prolonger sur $]-\infty, 0[$ en la supposant paire, afin de simplifier l'expression du développement en série entière

a_l étant les coefficients du développement en série entière : $a_l = a^{(l)}(0)/l!$. Et finalement l'expression du signal en sortie de l'amplificateur non linéaire est :

$$s_0(t) = \sum_{l=0}^{\infty} a_l |s(t)|^{2l} s(t) \tag{3.6}$$

On ne peut pas calculer tous les termes de cette somme, mais il est possible de calculer les premiers. Posons

$$\begin{aligned} s_0(t) &= \sum_{l=0}^{\infty} s_{0l}(t) \\ \text{où } s_{0l}(t) &= a_l |s(t)|^{2l} s(t) \end{aligned} \tag{3.7}$$

Les deux premiers termes ont pour expression :

$$\begin{cases} s_{00}(t) = a_0 s(t) = a(0)s(t) = \dfrac{\nu}{N_p} \displaystyle\sum_{j=0}^{N_p-1} c_j e^{2i\pi(f' + \frac{j}{T_S})t} \\ \\ s_{01}(t) = a_1 |s(t)|^2 s(t) = a'(0)|s(t)|^2 s(t) \end{cases} \tag{3.8}$$

La valeur de $|s(t)|^2$ est :

$$|s(t)|^2 = s(t)\overline{s(t)} = \frac{1}{N_p^2} \sum_{j=0}^{N_p-1} \sum_{j'=0}^{N_p-1} c_j \overline{c_{j'}} e^{2i\pi \frac{j-j'}{T_S} t} \tag{3.9}$$

Et donc :

$$s_{01}(t) = \frac{a_1}{N_p^3} \sum_{j=0}^{N_p-1} \sum_{j'=0}^{N_p-1} \sum_{j''=0}^{N_p-1} c_j \overline{c_{j'}} c_{j''} e^{2i\pi(f' + \frac{j-j'+j''}{T_S})t} \tag{3.10}$$

Et en appliquant le changement de variable $n = j - j' + j''$, on obtient :

$$s_{01}(t) = \frac{a_1}{N_p^3} \sum_{j=0}^{N_p-1} \sum_{j'=0}^{N_p-1} \sum_{n=j-j'}^{N_p-1+j-j'} c_j \overline{c_{j'}} c_{n-j+j'} e^{2i\pi(f' + \frac{n}{T_S})t} \tag{3.11}$$

On remarque tout d'abord que n varie de $-N_p + 1$ à $2N_p - 2$, suivant les valeurs de j et j'. Si $n < j - j'$ l'indice $n - j + j'$ devient inférieur à 0, et si $n \rangle N_p - 1 + j - j\prime$ il devient supérieur ou égal à N_p. Donc si l'on étend la définition de c_j avec des coefficients nuls :

$$c_j = \begin{cases} 0 & \text{si } j < 0 \\ \text{composantes du symbole OFDM} & \text{si } j \in [0, N_p - 1] \\ 0 & \text{si } j \geq N_p \end{cases} \tag{3.12}$$

On peut alors écrire :

$$s_{01}(t) = \frac{a_1}{N_p^3} \sum_{n=-N_p+1}^{2N_p-2} (\sum_{j=0}^{N_p-1} \sum_{j'=0}^{N_p-1} c_j \overline{c_{j'}} c_{n-j+j'}) e^{2i\pi(f' + \frac{n}{T_S})t} \tag{3.13}$$

Ce qui nous permet de regrouper les termes par fréquence porteuse. Les symboles reçus d_k sont calculés avec la transformée de Fourier de $s_0(t)$, et correspondent chacun à la porteuse à la fréquence $f' + k/T_S$. Comme pour $s(t)$, on peut décomposer chaque d_k en une somme :

$$\begin{cases} d_k = \sum_{l=0}^{\infty} d_{k,l} \\ \\ d_{k,l} = \sum_{j=0}^{N_p-1} s_{0l}(\frac{j}{N_p} T_S) e^{-2i\pi(f' + \frac{k}{T_S}) \frac{j}{N_p} T_S} \end{cases} \tag{3.14}$$

Chaque terme $d_{j,l}$ correspond à la contribution du terme s_{0l} de la décomposition de $s_0(t)$. Les valeurs de $d_{k,0}$ et $d_{k,1}$ se déduisent facilement des équations (3.8) et (3.13), puisqu'ils sont égaux à chaque fois au facteur de $e^{2i\pi(f'+k/T_S)t}$ dans $s_{0l}(t)$, multiplié par N_p :

$$\begin{aligned} d_{k,0} &= \nu c_k \\ \\ d_{k,1} &= \frac{a_1}{N_p^2} \sum_{j=0}^{N_p-1} \sum_{j'=0}^{N_p-1} c_j \overline{c_{j'}} c_{k-j+j'} \end{aligned} \tag{3.15}$$

Les termes $d_{k,0}$ sont proportionnels aux symboles émis sur les différentes porteuses, et correspondent à la composante linéaire du système OFDM. Les termes $d_{k,1}$ quand à eux sont proportionnels à une somme de produits 3 à 3 des symboles émis, et correspondent à l'intermodulation d'ordre 3. On peut montrer de même que les symboles $d_{k,2}$ sont proportionnels à une somme de produits 5 à 5 des symboles émis, et ainsi de suite.

Dans un système OFDM classique les termes $d_{k,l},\ l \geq 1$ sont des perturbations qui vont augmenter le taux d'erreur binaire de la transmission. Dans un système avec une compensation des non-linéarités dans le récepteur, un module doit pouvoir inverser la fonction $\mathbf{c} \rightarrow \mathbf{d}$, c'est à dire retrouver les symboles émis c_j à partir des symboles reçus d_k. Le calcul précédent montre que le symbole reçu sur une porteuse ne dépend pas uniquement du symbole émis sur cette même porteuse, mais également de produits entre les symboles émis sur les autres porteuses. Donc un réseau de neurones qui corrige la non-linéarité de l'amplificateur sur une porteuse doit prendre en compte ce qui est reçu sur toutes les porteuses, et en particulier les corrélations d'ordre supérieur entre ces porteuses.

1.3. Symétries de la non-linéarité dans le domaine fréquentiel

On a donc vu que le réseau de neurones a besoin des symboles reçus sur toutes les porteuses pour compenser la non-linéarité. Il faudrait donc choisir un réseau avec N_p entrées et N_p sorties, où N_p est le nombre de porteuses, dans le cas où le réseau de neurones est capable de traiter des informations complexes. Si le réseau ne traite que des informations réelles, il faut $2N_p$ entrées et sorties réelles.

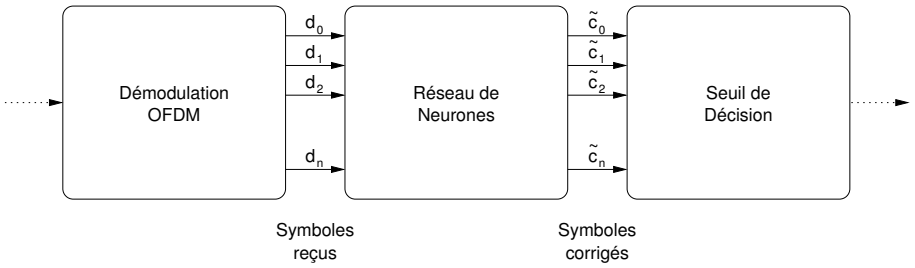

Figure 3.4. Schéma du récepteur OFDM avec un réseau de neurones complet

Cependant il est possible de chercher des symétries et des invariances dans le problème à résoudre, afin de simplifier cette structure. Tout d'abord on peut constater qu'il y a une invariance par décalage des porteuses. En effet si l'on crée le signal suivant :

$$s^{(n)}(t) = s(t)e^{2i\pi\frac{nt}{T_S}} \tag{3.16}$$

$s(t)$ étant défini dans (3.3). On constate alors que :

$$|s^{(n)}(t)| = |s(t)| \tag{3.17}$$

et donc :

$$s_0^{(n)}(t) = s_0(t)e^{2i\pi\frac{nt}{T_S}} \tag{3.18}$$

Ainsi lorsque l'on décale en fréquence les porteuses dans un système OFDM non linéaire, les porteuses reçues sont également décalées en fréquence, avec le même écart. Il est donc possible de créer un réseau de neurones qui ne possède qu'une sortie, et donc qui ne calcule le symbole corrigé que sur une porteuse. Les symboles corrigés sur les autres porteuses pourront être obtenus en décalant les entrées du réseau. Il faut donc ajouter des entrées au réseau de neurones afin de pouvoir réaliser ce décalage, décaler les entrées du rang voulu en fonction du symbole que l'on veut déterminer, et compléter les autres entrées à 0 :

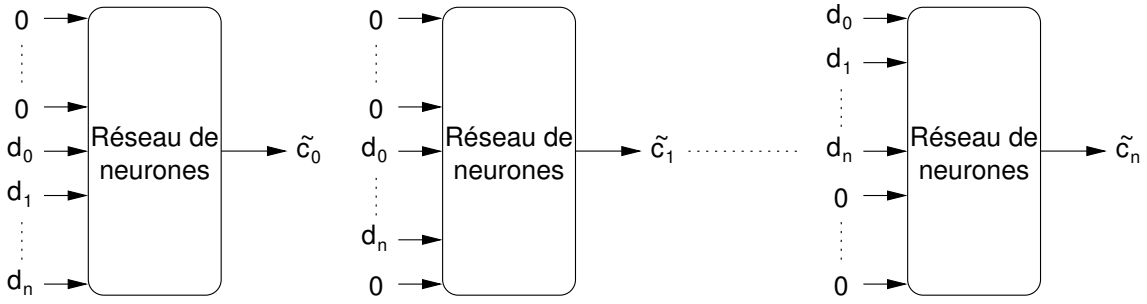

Figure 3.5. Emploi d'un réseau à une sortie pour calculer tous les symboles par décalage

Dans toutes les architectures testées, cette méthode permet de réduire de manière significative le nombre de poids, et donc d'augmenter la vitesse de convergence de l'apprentissage du réseau, et de réduire la puissance de calcul utilisée ainsi que le nombre d'exemples nécessaires dans la base. Dans le cas du réseau RPN (section 4.5 p. 63) le nombre de poids nécessaires du réseau est :

$$(N_e + 1)\frac{O_r(O_r + 1)}{2}N_s \tag{3.19}$$

N_e étant le nombre d'entrées, N_s celui du nombre de sorties, et O_r l'ordre du réseau. En effet un réseau RPN est composé de O_r réseaux pi-sigma, d'ordres $1 \dots O_r$ et dont le nombre de poids est donné dans l'équation (2.28). De plus pour réaliser N_s sorties il faut N_s réseaux RPN. Donc dans le premier cas le nombre de poids nécessaires pour le réseau est de :

$$(2N_p + 1)\frac{O_r(O_r + 1)}{2}2N_p = (2N_p + 1)N_pO_r(O_r + 1) \tag{3.20}$$

Et dans le second cas, avec décalage (on doit ajouter $2(N_p - 1)$ entrées et il n'y a plus qu'une sortie complexe, donc deux sorties réelles) :

$$(4N_p - 2)\frac{O_r(O_r + 1)}{2}2 = (4N_p - 2)O_r(O_r + 1) \tag{3.21}$$

Le nombre de poids a donc été divisé par presque $2N_p$ lorsque N_p est assez grand.

Maintenant si l'on suppose que deux symboles OFDM temporels $s(t)$ et $s_\delta(t)$ diffèrent uniquement par un retard δ, c'est à dire que :

$$s_\delta(t) = s(t - \delta) \tag{3.22}$$

alors on a à la réception :

$$s_{\delta 0}(t) = s_\delta(t)g(|s_\delta(t)|) = s(t - \delta)g(|s(t - \delta)|) = s_0(t - \delta) \tag{3.23}$$

Le système OFDM non linéaire est donc à temps invariant, c'est à dire que lorsqu'un retard est présent en entrée du système OFDM non linéaire, le signal en sortie aura le même retard. Si

l'on traduit cette propriété en fréquentiel, en notant respectivement $\mathbf{c}$ et $\mathbf{d}$ les symboles OFDM émis et reçu dans un cas, et $\mathbf{c}_\delta$ et $\mathbf{d}_\delta$ les symboles émis et reçus avec un retard δ :

$$\begin{aligned} c_{\delta,k} &= c_k e^{-2i\pi\frac{k}{T_S}\delta} \\ d_{\delta,k} &= d_k e^{-2i\pi\frac{k}{T_S}\delta} \end{aligned} \tag{3.24}$$

Avec le même raisonnement, on constate que si l'on applique une rotation aux symboles, c'est à dire que si on les multiplie tous par un complexe de module 1, par conservation du module de $s(t)$ on observera la même rotation dans les symboles reçus. C'est-à-dire, avec une notation similaire à la précédente :

$$\begin{aligned} c_{\alpha,k} &= c_k e^{i\alpha} \\ d_{\alpha,k} &= d_k e^{i\alpha} \end{aligned} \tag{3.25}$$

Les deux propriétés (3.24) et (3.25) sont vérifiées par le système OFDM non linéaire, et doivent être prises en considération pour faciliter l'apprentissage du réseau de neurones. Deux méthodes sont envisageables. Tout d'abord il est possible de modifier la structure du réseau de neurones afin qu'il vérifie ces propriétés de manière intrinsèque. L'article [GILE94] expose quelques idées pour construire des réseaux d'ordre supérieur présentant certaines invariances géométriques, mais les méthodes dépendent de l'architecture du réseau, et peuvent être complexes à mettre en oeuvre. Une autre méthode consiste à augmenter la taille de la base d'apprentissage. A partir d'un exemple de la base il est possible de créer d'autres exemples en appliquant des rotations et des retards. Ainsi on peut augmenter la taille de la base de plusieurs ordres de grandeurs sans pour autant augmenter le nombre de mesures faites sur un système OFDM. C'est cette méthode qui a été choisie, en raison de la simplicité de sa mise en oeuvre ainsi que son indépendance vis-à-vis de l'architecture du réseau de neurones utilisé.

2. Mise en oeuvre et protocole expérimental

2.1. Simulation du système OFDM

Pour simuler un système OFDM, l'outil SPW (Signal Processing Workshop) de Cadence a été choisi en raison de ses performances au niveau du temps de calcul et de sa souplesse. Deux modèles d'amplificateurs ont été employés dans les différentes simulations. L'amplificateur SSPA (présenté en p. 66) qui représente bien les amplificateurs à semi-conducteurs utilisé en radio mobile, et le limiteur (présenté en p. 30), qui lui représente bien les amplificateurs radio munis d'une prédistorsion ou ceux utilisés pour les liaisons filaires de type ADSL.

Le modèle SSPA comporte un paramètre p qui permet de régler la forme de la réponse de l'amplificateur. Nous avons utilisé les valeurs 2 et 3 dans nos simulations, ce qui correspond le plus aux amplificateurs réels [RAPP91]. La courbe figure 3.6 montre la réponse d'un tel amplificateur, en représentant le module de l'entrée en abscisse et celui de la sortie en ordonnée. Les courbes pour $p = 2$ et $p = 3$ sont montrées, ainsi que celle pour $p = \infty$, ce qui correspond au modèle du limiteur. La non-linéarité de phase n'est pas représentée ici, car avec le modèle SSPA elle est nulle, et ainsi aucun déphasage n'est introduit par la non-linéarité de l'amplificateur.

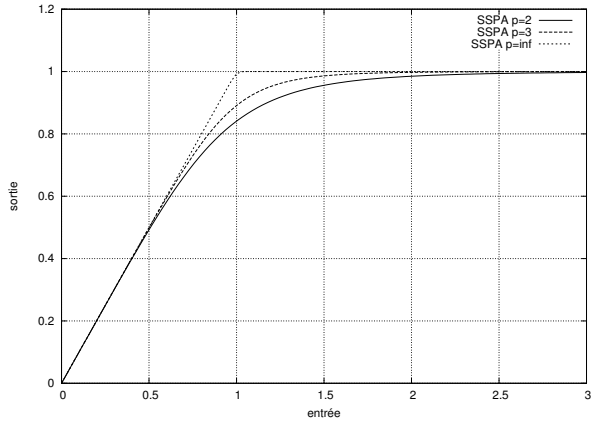

Figure 3.6. Réponse du modèle SSPA pour différentes valeurs de p

On rappelle que la sévérité de la non-linéarité peut être représentée par le recul, ou Input Back Off en anglais (p. 30) et qui est le rapport entre la puissance de saturation ramenée à l'entrée et la puissance moyenne du signal. Plus le recul est élevé, moins les distorsions linéaires sont importantes.

Dans un premier temps pour tester les différentes architectures neuronales un canal très simple a été simulé. Il s'agit d'un canal gaussien, qui n'est pas sélectif en fréquence. Dans ce canal le signal est propagé sans distorsion, et un bruit blanc additif gaussien est ajouté. Le rapport signal sur bruit (RSB ou SNR pour Signal to Noise Ratio en anglais) est une mesure relative du bruit, égale (à un logarithme près) au rapport entre la puissance moyenne du signal et celle du bruit blanc gaussien. Cependant dans le domaine des télécommunications ce qui est le plus significatif pour le choix d'une architecture ce n'est pas simplement la puissance d'émission du signal, mais plutôt l'énergie dépensée par unité de quantité d'information. Un autre critère est donc utilisé, noté E_b/N_0 et défini p. 33. On rappelle que sa valeur est le rapport entre l'énergie dépensée pour émettre un bit d'information et la densité spectrale du bruit. Il est donc fonction du rapport signal sur bruit, mais également du rendement du codage canal et de la modulation. De plus il est proportionnel au RSB, et lorsqu'il est exprimé en dB, est égal au RSB ajouté d'un facteur que l'on peut calculer à l'aide du rendement du code et du nombre de bits par symbole.

2.2. Constitution d'une base et apprentissage du réseau de neurones

Pour constituer une base d'apprentissage, on se sert d'une simulation du système OFDM comprenant le canal ainsi que son égalisation. Un générateur aléatoire produit des informations binaires, et l'on stocke pour chaque symbole OFDM simulé le symbole émis et le symbole reçu. Les symboles reçus seront fournis comme entrées du réseau de neurones lors de l'apprentissage, et le symbole émis sera donné comme réponse voulue. Le réseau va ainsi apprendre à inverser la non-linéarité.

Plusieurs paramètres sont importants pour la génération de la base d'apprentissage. Tout d'abord les caractéristiques de la modulation OFDM : en effet les paramètres tels que le nombre de porteuses, le codage binaire à symbole ainsi que le recul et le modèle de l'amplificateur peuvent tous modifier le comportement fréquentiel de la non-linéarité, et donc la fonction que doit réaliser le réseau de neurones. Dans les présentations de résultats, ces paramètres

seront systématiquement indiqués. Ensuite le nombre d'exemples dans la base d'apprentissage est également très significatif. Trop peu d'exemples et l'on risque un surapprentissage du réseau, trop d'exemples et la quantité de mémoire et le temps nécessaire pour créer la base d'apprentissage deviennent prohibitifs.

Enfin un dernier paramètre important est la puissance de bruit du canal. En effet lorsqu'il est utilisé dans la chaîne de communication OFDM, le canal ajoute du bruit au signal, et donc le réseau de neurones doit travailler avec un signal bruité en entrée. Pour son apprentissage, on a donc le choix entre donner au réseau une base sans bruit, ou bien avec plus ou moins de bruit que lors de son exploitation. On peut supposer qu'avec une base sans bruit le réseau apprend plus facilement la fonction à réaliser. Cependant si l'on veut qu'il ait une grande immunité au bruit, il est nécessaire de lui faire apprendre avec du bruit. On peut s'en rendre compte sur un exemple simple : supposons que l'on soit en présence d'un canal parfait, sans aucune distorsion ni linéaire ni non linéaire. On veut apprendre à un réseau un décodage de type MAQ4, et en particulier celui qui va décoder la partie réelle du signal. Le réseau doit fournir -1 en sortie si la partie réelle du signal est négative, et +1 si elle est positive. Ce système peut être réalisé très simplement sans réseau de neurones à l'aide d'un seuil de décision, mais nous allons étudier ce réseau afin de comprendre l'influence du bruit. La figure 3.7 montre la modulation MAQ4 avec un bon rapport signal sur bruit ($20dB$) et la réponse du réseau de type sigmoïde qui pourrait réaliser la fonction voulue.

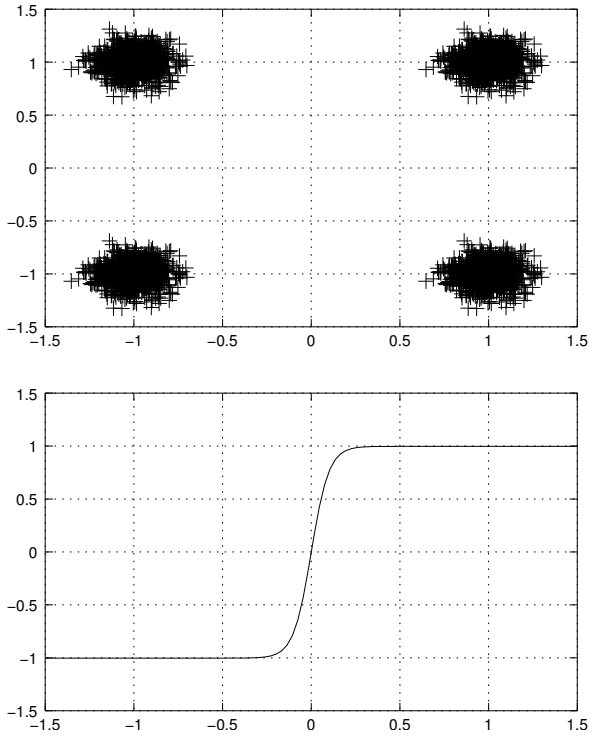

Figure 3.7. Modulation MAQ4 avec un RSB de 20 dB et sigmoïde idéale associée

On constate cependant qu'il n'y a aucun exemple dans la base d'apprentissage dont l'abscisse est comprise entre -0.6 et 0.6. L'algorithme d'apprentissage peut donc aboutir a des réseaux qui réalisent des fonctions légèrement différentes, avec une sigmoïde ayant une pente plus douce, ou une sigmoïde décalée par exemple comme dans la figure 3.8 :

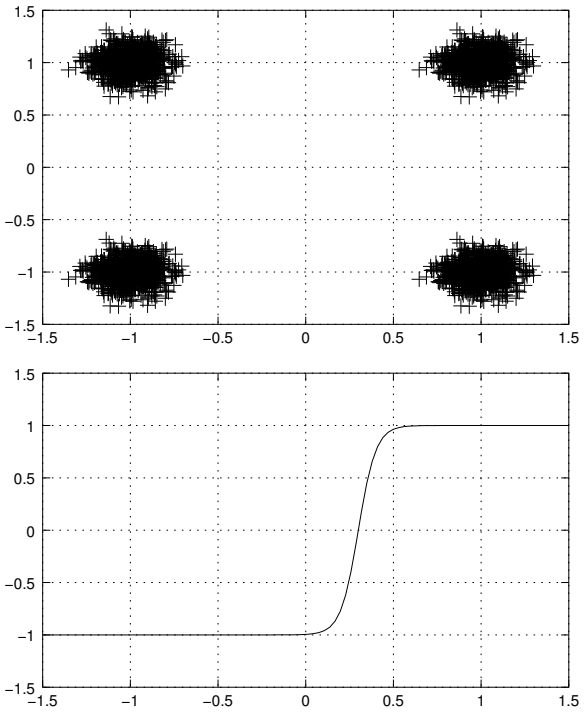

Figure 3.8. Sigmoïde décalée dans les mêmes conditions que la figure précédente

Théoriquement cette solution n'est pas optimale, même avec cette base d'apprentissage, car la sigmoïde n'atteint jamais exactement les valeurs -1 et 1, et l'erreur quadratique moyenne à la fin de l'apprentissage est légèrement plus élevée dans ce cas que le précédent, figure 3.7. Cependant en pratique avec des algorithmes tels que la descente de gradient il est possible d'obtenir un tel résultat. Le réseau ainsi formé aura aussi une erreur d'apprentissage très faible, avec des valeurs en sortie d'environ -1 pour les exemples de la base situés à gauche et 1 pour ceux situés à droite. Supposons que le réseau soit ensuite confronté à des données avec un rapport signal sur bruit inférieur, comme par exemple $3dB$ dans l'exemple suivant, figure 3.9. Du fait du décalage de la sigmoïde, pour des symboles qui auraient dû provoquer une sortie à 1, le réseau va en fait donner en sortie une valeur entre -1 et 1. L'erreur du réseau en utilisation sera donc plus élevée :

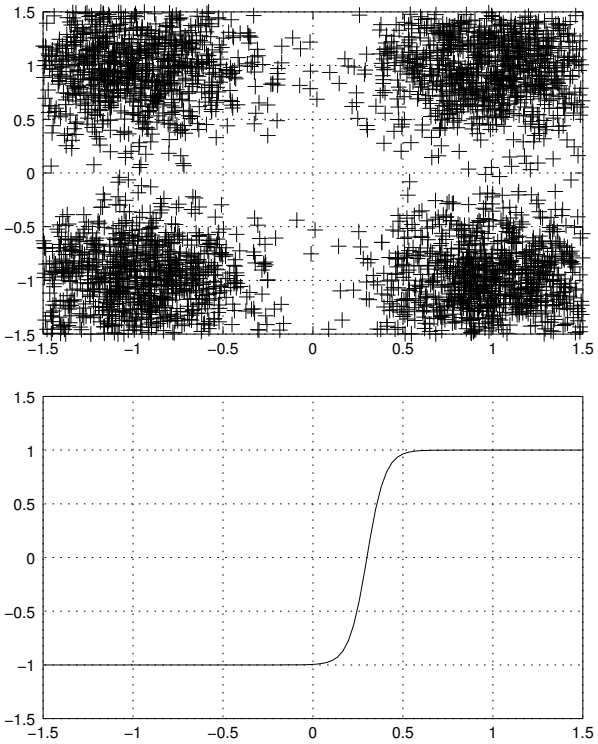

Figure 3.9. Données avec un bruit de 3 dB avec la fonction sigmoïde obtenue figure précédente

Si l'apprentissage du réseau avait été fait avec la base ayant un RSB de $3dB$ au lieu de $20dB$, l'algorithme aurait sélectionné une sigmoïde plus centrée, car celle-ci aurait engendré une erreur d'apprentissage plus faible que la sigmoïde décalée que l'on voit ci-dessus. On en déduit donc qu'il faudrait que la base d'apprentissage soit constituée avec un bruit au moins aussi puissant que celui qui sera présent lors de l'utilisation du réseau de neurones. Ceci sera également vérifié expérimentalement.

Une fois la base d'apprentissage constituée, différentes architectures de réseaux de neurones sont testées et comparées. La fonction d'activation de la dernière couche des réseaux doit cependant être adaptée à la modulation que l'on désire. Par exemple la sigmoïde est adaptée aux modulations de type MAQ4, comme on vient de l'illustrer dans les figures précédentes. Pour une modulation plus complexe, comme une MAQ16, il est possible de choisir une sigmoïde présentant deux plateaux supplémentaires, qui permettront de guider le réseau vers les 4 sorties différentes que l'on désire qu'il produise. Par exemple :

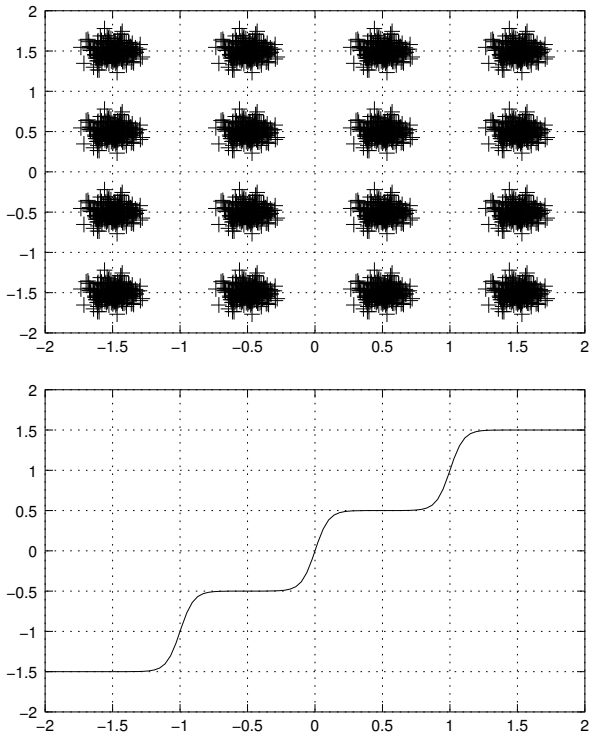

Figure 3.10. Exemple de fonction d'activation adaptée à une modulation MAQ16, avec un rapport signal sur bruit de 20 dB

De telles fonctions d'activation ont déjà été crées et étudiées dans le contexte des télécommunications et des travaux montrent qu'elles sont bien adaptées à ces modulations [MAN94], [BALA95].

Dans les exemples précédents la fonction que doit réaliser le réseau de neurones est évidente, ce qui permet d'illustrer facilement le problème posé par le bruit et l'adaptation de la fonction d'activation à la modulation. En pratique sur un canal multiporteuses avec une non-linéarité la tâche que devra réaliser le réseau est bien plus complexe. En effet les non-linéarités vont induire des perturbations entre les différentes porteuses (voir section 1.2 p. 66) et la séparation entre les différents symboles de la constellation ne sera pas si évidente à accomplir.

2.3. Résultats et interprétations

L'erreur du réseau de neurones à la fin de son apprentissage peut nous renseigner sur ses capacités à réduire les effets des non linéarités, mais le seul critère objectif qui nous permet de mesurer ses performances est le taux d'erreur binaire dans une chaîne de transmission OFDM complète. Donc une fois l'apprentissage terminé, on simule à nouveau un système OFDM avec le correcteur, et l'on mesure le taux d'erreur binaire (rapport entre le nombre de bits erronés et le nombre de bits transmis) de la transmission en fonction du rapport signal sur bruit (E_b/N_0). Ces résultats sont présentés sous forme d'une courbe avec le taux d'erreur binaire en ordonnée et le rapport signal sur bruit en abscisse. Une courbe seule n'est pas très utile en elle-même, mais c'est la comparaison entre deux courbes qui est intéressante. La figure 3.11 montre un exemple de résultat obtenu avec un correcteur qui sera présenté plus tard dans cette thèse :

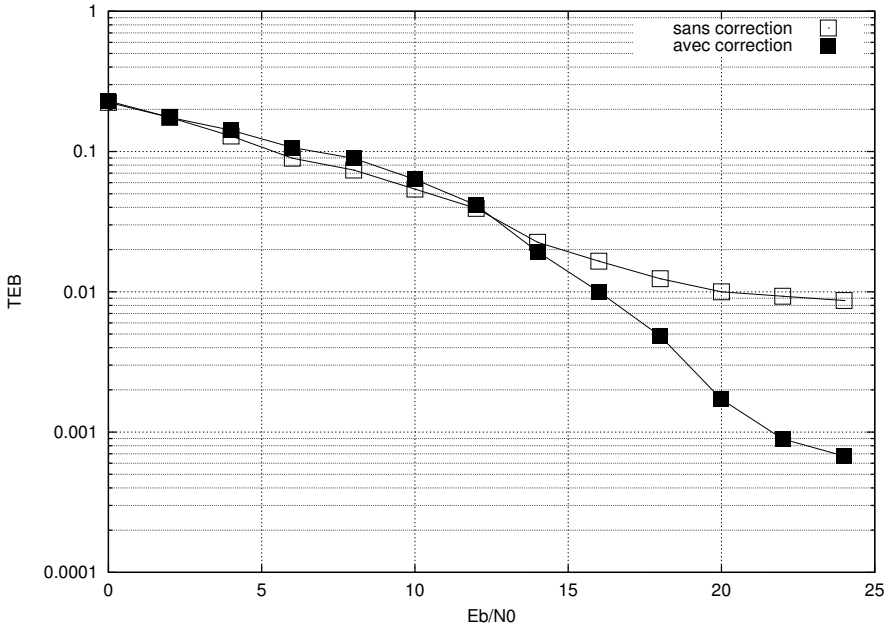

Figure 3.11. Exemple de résultat de correcteur

Le système comporte 48 porteuses, avec une modulation MAQ16 et un recul de l'amplificateur de 0 dB. Le taux d'erreur est plus faible avec le correcteur, celui-ci réduit donc les effets des non-linéarités, à partir d'un certain rapport signal sur bruit. Cette courbe peut être exploitée de deux façons. On peut se fixer un rapport signal sur bruit, et quantifier le gain sur le taux. Par exemple à 20 dB, le nombre d'erreurs sur la transmission est divisé par 5. Mais une autre interprétation plus intéressante est faite à taux d'erreur binaire constant. En effet en pratique le taux d'erreur voulu sur une transmission est fixée par l'application : on veut pouvoir garantir un certain taux. Si l'on se fixe un objectif de 10^{-2} pour le taux d'erreur binaire, on constate une différence de 4 dB sur le rapport signal sur bruit. Ceci veut dire que le correcteur permet d'utiliser un signal avec une puissance inférieure de 4 dB, avec les mêmes performances. On peut donc diviser la puissance de l'amplificateur par un facteur de plus de 2,5 grâce au correcteur, sans changer la qualité de la transmission. Cet aspect est le plus intéressant pour une application donnée, car un amplificateur moins puissant coûte moins cher et consomme moins d'énergie. La valeur du taux d'erreur binaire choisie pour cette comparaison, 10^{-2}, correspond souvent au point à partir duquel le code canal commence à être efficace en terme de correction d'erreurs, et est donc un point intéressant pour l'étude de la compensation des effets des non-linéarités.

3. Performances d'un correcteur basé sur un PMC

3.1. Introduction

Un premier essai a été fait avec un réseau de neurones classique. Les PMC (perceptrons multicouches, présentés en section 3 p. 53) ont été choisis plutôt que les GRBF en raison des caractéristiques du système. En effet l'espace d'entrée du réseau est multidimensionnel, le nombre de dimensions étant égal au nombre de porteuses dans le cas d'un codage binaire à symbole réel, ou au double du nombre de porteuses dans le cas d'un codage binaire à symbole complexe. Les données sont réparties de manière uniforme dans chaque dimension, et donc

un réseau GRBF nécessiterait un grand nombre de prototypes pour paver l'espace d'entrée, nombre qui augmenterait exponentiellement avec le nombre de porteuses.

Un PMC réalise une approximation plus globale, et doit pouvoir donc s'accommoder plus facilement d'un nombre de porteuses plus élevé. Trouver le perceptron le plus adapté au problème est une tâche difficile, et qui ne peut être faite qu'expérimentalement. Il ne serait pas intéressant de détailler toutes les simulations qui ont été faites, mais les plus significatives sont présentées dans la suite.

Les simulations ont tout d'abord été faites sur un système OFDM simple : le nombre de porteuses est de 4, la modulation utilisée pour le codage binaire est une MAQ16, et aucun bruit n'est présent sur le canal. Les apprentissages ont été effectués avec un algorithme de descente de gradient (voir en annexe, section 1 p. 137), qui nécessite de spécifier un coefficient réglant la vitesse de convergence. La valeur de ce coefficient a également été déterminée expérimentalement : une valeur trop faible et la convergence est très lente, une valeur trop importante et l'apprentissage est instable.

Cet apprentissage est effectué à l'aide du gradient total, c'est à dire que lors d'une itération, le gradient est tout d'abord calculé a l'aide de tous les exemples de la base d'apprentissage, puis ensuite les nouveaux poids sont déterminés. Une base de validation est également crée avec le même nombre d'éléments que dans la base d'apprentissage, mais des éléments différents de cette dernière, afin de vérifier que le réseau ne fait pas de surapprentissage. Nous présentons des courbes d'apprentissage, qui montrent l'évolution de la performance des PMC (l'erreur quadratique moyenne) sur les bases d'apprentissage et de validation, au fur et à mesure de l'apprentissage. En abscisse nous avons le numéro de l'itération. Tous les apprentissages ont été réalisés chacun 10 fois. En effet comme l'initialisation des poids du réseau de neurones est faite de manière aléatoire, il faut s'assurer que le profil de l'apprentissage (et principalement l'erreur quadratique moyenne à la fin de l'apprentissage) ne dépend pas de l'initialisation.

Bien que la théorie ait déterminé qu'un PMC avec une seule couche cachée est un approximateur universel et donc pourrait convenir à cette tâche, nous avons principalement testé des PMC à deux couches cachées. En effet en pratique ces perceptrons sont plus efficaces dans de nombreux cas, c'est à dire qu'ils peuvent réaliser des fonctions plus complexes avec moins de paramètres. Par contre l'apprentissage de ces réseaux est généralement plus difficile.

3.2. Architecture

Différentes architectures de PMC ont été employées, en augmentant progressivement le nombre de neurones dans la couche cachée, puis en ajoutant une seconde couche cachée. Des bases de différentes tailles ont également été utilisées. Cependant le temps prohibitif d'apprentissage (jusqu'à 70 heures comme on le verra dans la suite, qu'il faut encore multiplier par 10 car chaque architecture est testée plusieurs fois) ne nous permet pas une étude extrêmement poussée en faisant varier finement tous les paramètres. Nous nous sommes donc limités à quelques bases d'apprentissage et quelques architectures de PMC. Pour simplifier les notations nous noterons 'perceptron-n' un PMC à une couche cachée avec n neurones, et 'perceptron-n-m' un PMC à deux couches cachées constituées respectivement de n et m neurones.

Le critère d'arrêt de l'apprentissage est ici le nombre d'itérations[8] , que l'on a limité à 2000. La figure 3.12 montre la courbe d'apprentissage d'un perceptron-10 sur une base de 512 symboles OFDM. On ne peut pas véritablement parler de convergence de l'algorithme car l'erreur quadratique moyenne du réseau continue de décroître au bout de 2000 itérations. Des mesures complémentaires avec un nombre d'itérations bien supérieur seront présentées dans la section suivante.

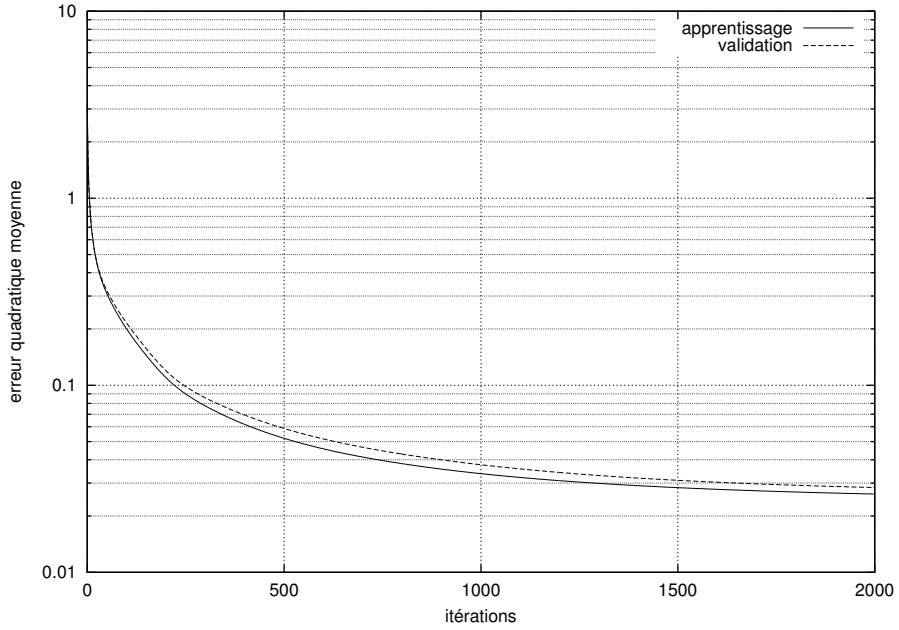

Figure 3.12. Apprentissage d'un perceptron-10 dans un système OFDM à 4 porteuses avec un amplificateur SSPA, une modulation MAQ16, un recul de 0 dB et avec une base de 512 éléments

Le même phénomène a été constaté avec des perceptrons ayant un nombre de neurones supérieurs sur la première couche, ainsi qu'avec des PMC à deux couches cachées, comme on le montre figure 3.13 dans le cas d'un perceptron-10-10 :

[8]les critères d'arrêt utilisés couramment sont plutôt un arrêt de l'évolution des poids, ou de la performance d'apprentissage, ou encore une augmentation de l'erreur quadratique moyenne de validation. Comme aucun de ces phénomènes n'a pu être constaté dans nos simulations, nous avons choisi un nombre d'itérations fixe.

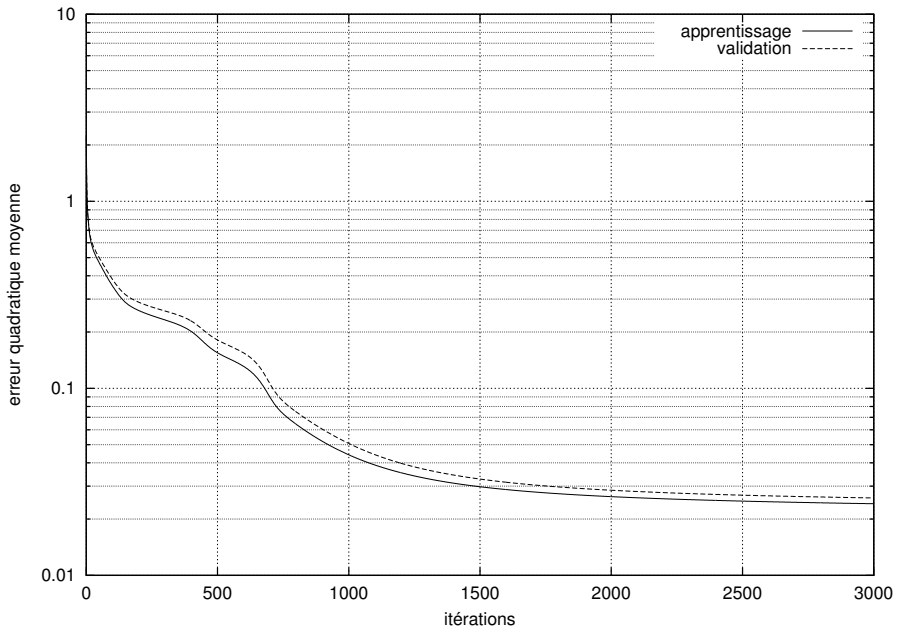

Figure 3.13. Apprentissage d'un perceptron 10-10 dans un système OFDM à 4 porteuses avec un amplificateur SSPA, une modulation MAQ16, un recul de 0 dB et avec une base de 512 éléments

La convergence est plus lente, à cause de l'ajout d'une couche cachée. C'est pour cette raison que le nombre d'itérations avant l'arrêt de l'algorithme d'apprentissage a été augmenté à 3000. On constate une infime diminution de l'erreur quadratique moyenne par rapport au perceptron-10, mais là encore la pente des courbes de performance est négative à la fin de l'apprentissage. Ce phénomène a été constaté sur tous les PMC que nous avons testé.

3.3. Apprentissage et convergence

Les deux courbes d'apprentissage présentées figures 3.12 et 3.13 laissent à penser qu'on devrait pouvoir obtenir une meilleure performance du réseau en augmentant le nombre d'itérations. Un essai a été fait sur un perceptron-30-25, sur une base de 2048 exemples, en portant le nombre d'itérations à 50000 :

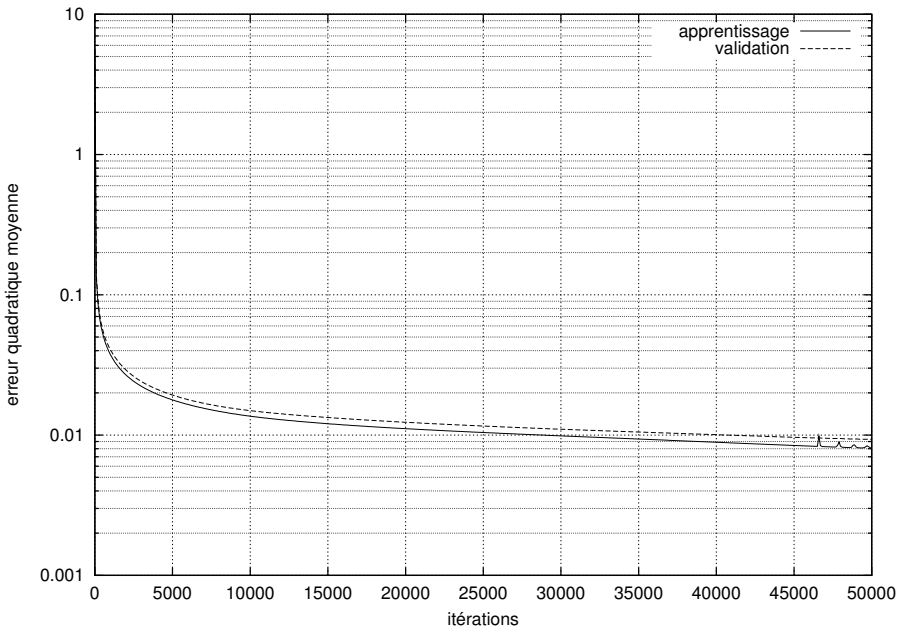

Figure 3.14. Apprentissage d'un perceptron-30-25 dans un système OFDM à 4 porteuses avec un amplificateur SSPA, une modulation MAQ16, un recul de 0 dB, une base de 2048 éléments et un plus grand nombre d'itérations

Sur cet essai l'erreur quadratique moyenne après 50000 itérations est de $8,08 \cdot 10^{-3}$ sur la base d'apprentissage et $9,32 \cdot 10^{-3}$ sur la base de validation. La performance est bien meilleure qu'avec seulement 2000 itérations, et la pente de la courbe laisse à penser qu'elle serait encore meilleure si l'on augmentait encore le nombre d'itérations.

Une comparaison des architectures avec un apprentissage de 50000 itérations serait intéressante, mais n'a pas été faite pour deux raisons : tout d'abord le temps de calcul est plutôt élevé, plus de 24 heures par apprentissage; comme il est nécessaire de faire plusieurs apprentissages sur chaque architecture afin de vérifier l'indépendance du résultat vis à vis de l'initialisation, chaque architecture de PMC demande environ une semaine de calcul. La seconde raison est que rien n'indique que 50000 itérations est en fait suffisant, il est possible que l'erreur quadratique moyenne se réduise encore d'un facteur non négligeable si l'ont continue la descente de gradient. Il est donc sans doute plus utile de travailler sur l'algorithme de convergence lui-même.

Dans une descente de gradient le coefficient qui est utilisé pour la mise à jour des paramètres est déterminant pour la rapidité de convergence et la stabilité de l'algorithme. Dans ces expériences, nous avons déterminé qu'un facteur 0,25 était le plus grand possible sans nuire à la stabilité de l'apprentissage. Il est toutefois possible que durant certaines phases de l'apprentissage, un coefficient plus important puisse permettre d'accélérer la convergence. Nous avons donc fait un apprentissage avec une descente de gradient adaptative (voir en annexe à la fin de la section 1 p. 137 sur la descente de gradient). Voici la courbe d'apprentissage d'un perceptron 30-25 avec l'algorithme adaptatif :

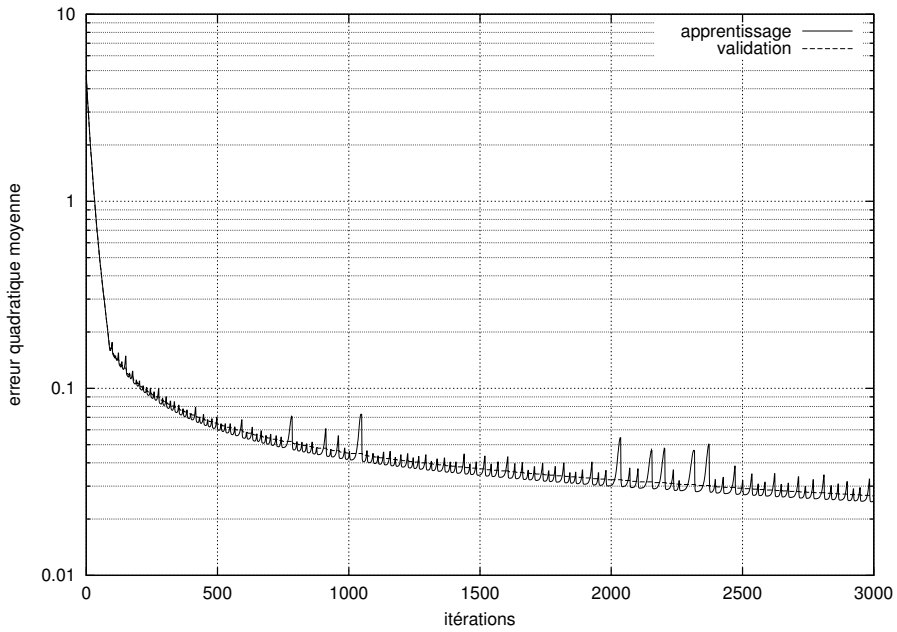

Figure 3.15. Apprentissage adaptatif d'un perceptron-30-25 dans un système OFDM à 4 porteuses avec un amplificateur SSPA, une modulation MAQ16, un recul de 0 dB et une base de 2048 éléments

On ne remarque pas d'amélioration de l'apprentissage avec un algorithme adaptatif, l'erreur quadratique moyenne à l'itération 3000 étant de $2,51 \cdot 10^{-2}$ sur la base d'apprentissage et $2,66 \cdot 10^{-2}$ sur la base de validation. Une autre solution est de choisir un algorithme d'optimisation d'ordre 2, comme Levenberg Marquardt (voir en annexe la section 2 p. 139) qui est couramment utilisé avec des réseaux de neurones. Les algorithmes d'ordre 2 demandent généralement moins d'itérations, mais chaque itération demande beaucoup plus de puissance de calcul et de mémoire, et ces demandes augmentent très vite avec le nombre de paramètres du système. L'apprentissage suivant, de 200 itérations sur un perceptron-30-25 sur une base de 2048 éléments a demandé plus de 70 heures de calcul et près de 800Mo de mémoire :

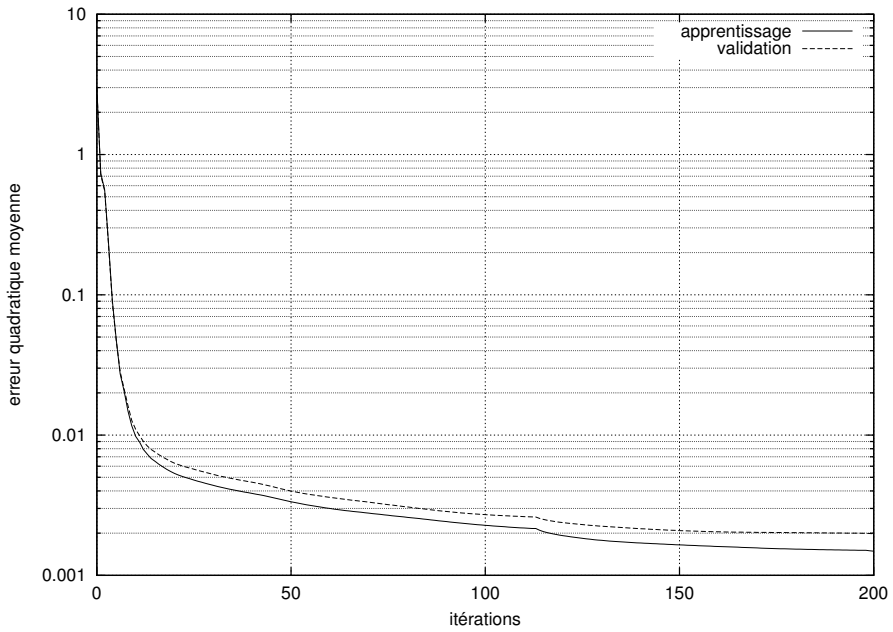

Figure 3.16. Apprentissage d'un perceptron-30-25 dans un système OFDM à 4 porteuses avec un amplificateur SSPA, une modulation MAQ16, un recul de 0 dB, une base de 2048 éléments et un algorithme de Levenberg Marquardt

On remarque une amélioration, les performances à l'itération 200 étant de $1,49 \cdot 10^{-3}$ sur la base d'apprentissage et de $1,98 \cdot 10^{-3}$ sur la base de validation. Cependant ici encore la pente des courbes de performance est négative à la fin de l'apprentissage. Il n'a pas été possible de tester toutes les architectures pour des raisons de temps de calcul et surtout de mémoire disponible, mais il ces résultats sont probablement transposables aux autres architectures de PMC.

3.4. Simulations et résultats

Pour évaluer les performances réelles du correcteur, il a ensuite été simulé dans un système OFDM (comprenant l'émetteur avec amplificateur SSPA, le canal gaussien, et le récepteur) avec différentes architectures de PMC. Quelques résultats sont présentés figure 3.17. La courbe 'ref' correspond au système de référence, sans correcteur mais avec la non-linéarité. Ensuite les correcteurs présentés utilisent un perceptron-10 ('p1'), un perceptron-30-25 après 2000 itérations d'apprentissage ('p2'), un perceptron-30-25 après 50000 itérations ('p3'), un perceptron-65-45 ('p4') et enfin un perceptron-30-25 avec un apprentissage par l'algorithme de Levenberg Marquardt ('p5').

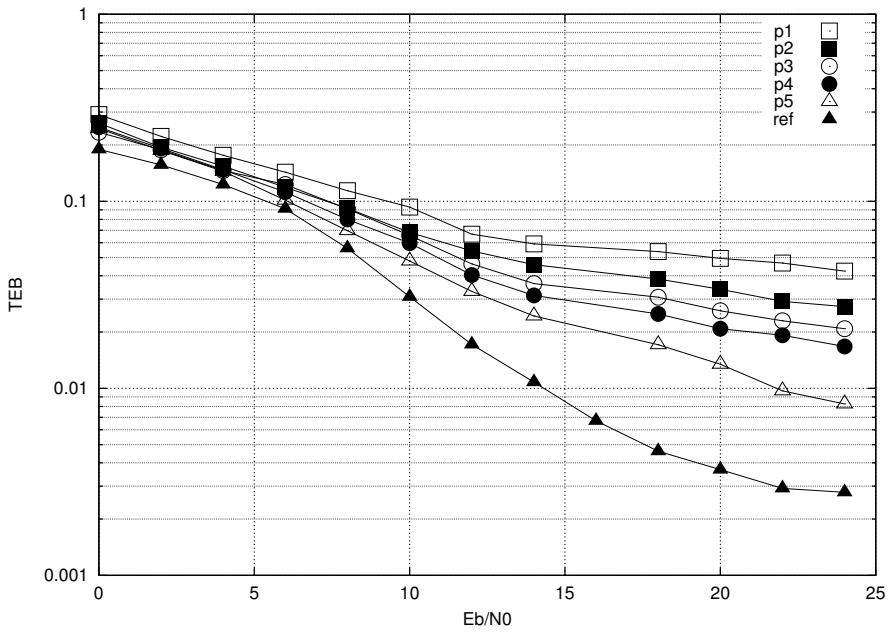

Figure 3.17. Taux d'erreur binaire d'un système OFDM à 4 porteuses avec une modulation MAQ16, un amplificateur SSPA, un recul de 0 dB et un correcteur à PMC (apprentissage sans bruit)

On constate qu'aucun perceptron ne permet de réduire le taux d'erreur par rapport à un système sans correcteur. La même chose a été constatée si l'on fait apprendre les PMC avec une base bruitée (figure 3.18). La base d'apprentissage a été créée avec un rapport signal sur bruit de 13 dB. Une discussion sur le choix de ce rapport signal sur bruit aura lieu dans la section suivante.

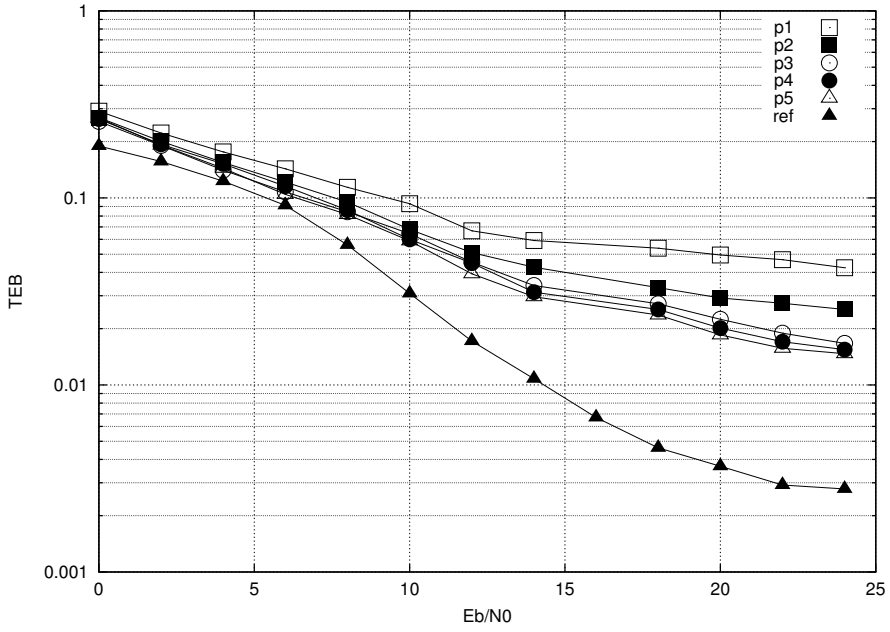

Figure 3.18. Taux d'erreur binaire d'un système OFDM à 4 porteuses avec une modulation MAQ16, un amplificateur SSPA, un recul de 0 dB et un correcteur à PMC (apprentissage avec bruit : E_b/N_0=13 dB).

Suite à ces résultats surprenants, un perceptron-8 sans apprentissage a été créé, avec des poids fixés de telle manière que sa sortie soit égale à l'entrée. La simulation a bien donné une courbe

confondue avec la courbe 'ref'. Le fait que l'on puisse créer un PMC en spécifiant les poids qui a de meilleures performances que tous les autres perceptrons prouve que le problème se situe au niveau de l'apprentissage. L'algorithme n'arrive pas à converger vers un jeu de paramètres correct pour le perceptron, et ceci pour toutes les architectures testées. Deux orientations sont alors possibles pour trouver une solution à ce problème. La première est d'essayer d'autres algorithmes d'apprentissage, ou d'utiliser des informations à priori sur le problème (telles que les règles de symétrie) pour aider l'apprentissage du réseau. La seconde est d'essayer un autre modèle de réseau de neurones. Cette dernière a été choisie, car les intermodulations qui apparaissent entre les porteuses sont des interférences d'ordre supérieur, ce qui laisse à penser que des réseaux d'ordre supérieur sont mieux adaptés au problème que les PMC.

4. Performances d'un correcteur basé sur un RPN

Différents réseaux d'ordre supérieur ont été simulés afin de réaliser un correcteur. Le réseau HPU (section 4.3 p. 60) a été écarté en raison du grand nombre de dimensions de l'espace d'entrée, ce qui aurait engendré un très grand nombre de poids. Les réseaux Square-MLP (section 4.2 p. 59) et Pi-Sigma (section 4.4 p. 61) n'ont pas permis d'obtenir de résultats plus satisfaisants que les PMC, mais par contre le RPN (section 4.5 p. 63) a montré les résultats plus intéressants, et c'est donc un correcteur basé sur cette architecture qui va être présenté.

4.1. Apprentissage et résultats du réseau RPN sur une base sans bruit

Les conditions sont les mêmes que pour le correcteur à base de perceptron. Nous avons utilisé la même base d'apprentissage : système OFDM à 4 porteuses avec une modulation MAQ16, une non-linéarité de type SSPA avec le paramètre p=2 et aucun bruit sur le canal. Dans un premier temps une base d'apprentissage de 256 symboles OFDM a été utilisée. La figure 3.19 montre l'évolution de la performance du RPN sur les bases d'apprentissage et de validation. L'algorithme d'optimisation est ici une descente de gradient adaptative, avec 200 itérations sur chaque ordre. On remarque une augmentation très élevée de l'erreur quadratique moyenne toutes les 200 itérations, ceci correspond à l'ajout au RPN d'un nouveau réseau pi-sigma, avec des poids initialisés aléatoirement. L'apprentissage d'un réseau RPN s'effectue en effet en plusieurs étapes successives, comme ceci était présenté dans le paragraphe 4.5 p. 63.

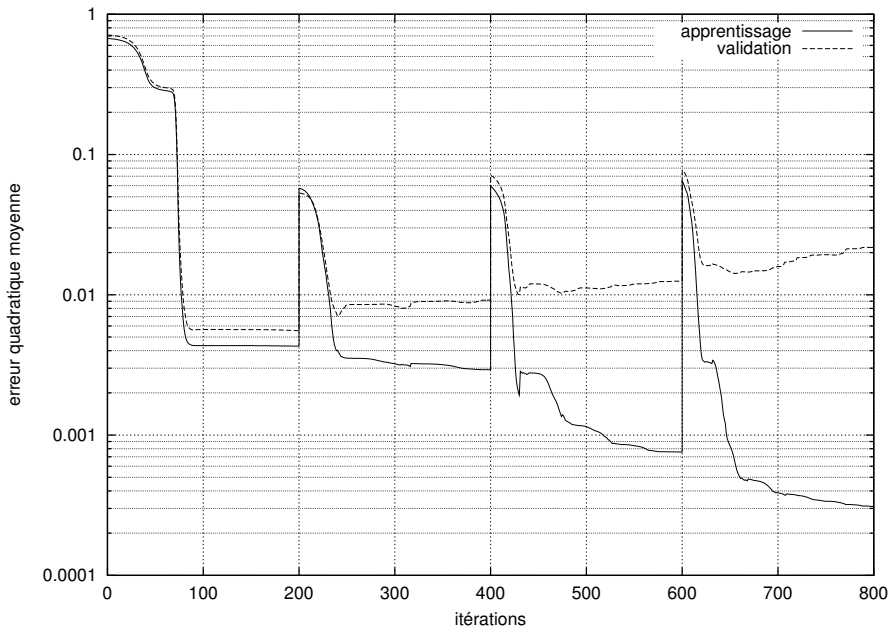

Figure 3.19. Apprentissage d'un RPN dans un système OFDM à 4 porteuses avec un amplificateur SSPA, une modulation MAQ16, un recul de 0 dB et une base de 256 éléments sans bruit

L'apprentissage a ici été stoppé à l'ordre 4. La courbe montre de manière évidente qu'il y a un surapprentissage, puisque l'erreur quadratique moyenne sur la base d'apprentissage diminue très rapidement, tandis que celle sur la base de validation augmente. Pour résoudre ce problème, une première méthode est de réduire la complexité du réseau. Cependant dans un réseau RPN ceci n'est pas possible, son architecture étant fixe. Le nombre de poids dépend simplement du nombre d'entrées et de sorties, ainsi que de l'ordre. La méthode qui a été plutôt choisie ici consiste à augmenter la taille de la base afin de permettre une meilleure généralisation du réseau. Nous l'avons augmentée progressivement, jusqu'à obtenir des résultats satisfaisants avec 16384 éléments :

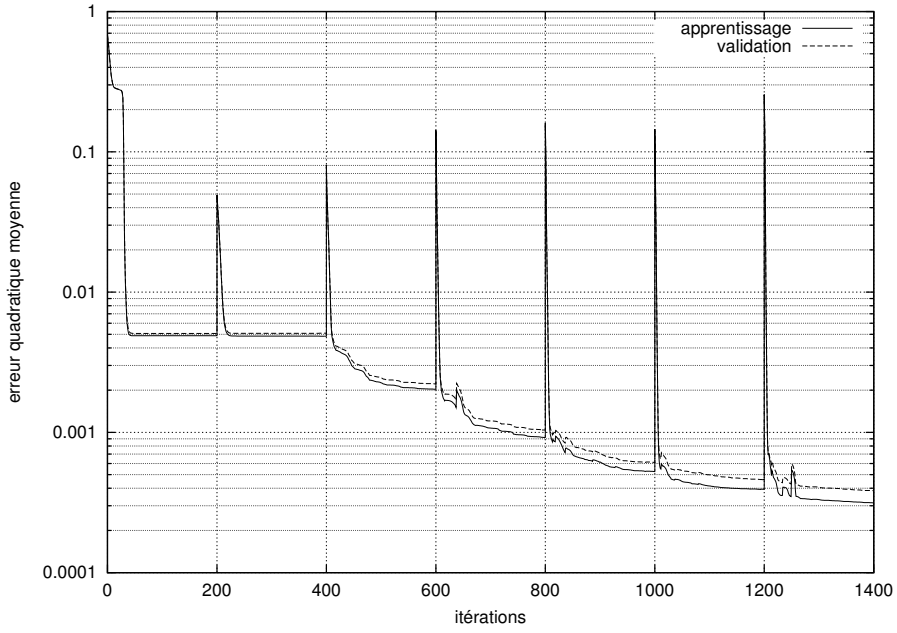

Figure 3.20. Apprentissage d'un RPN dans un système OFDM à 4 porteuses avec un amplificateur SSPA, une modulation MAQ16, un recul de 0 dB et une base de 16384 éléments sans bruit

Cet apprentissage a demandé 1 heure et 50 minutes sur une station Sun Ultra 10 avec l'algorithme de descente de gradient. Pour l'accélérer nous avons utilisé l'algorithme de Levenberg Marquardt, qui peut converger en moins d'itérations, mais avec plus de calcul sur chaque itération. Sur la même base, nous avons pu réduire le nombre d'itérations à 20 par ordre. L'apprentissage a duré 14 minutes sur la même station Sun et la courbe est montrée figure 3.21 :

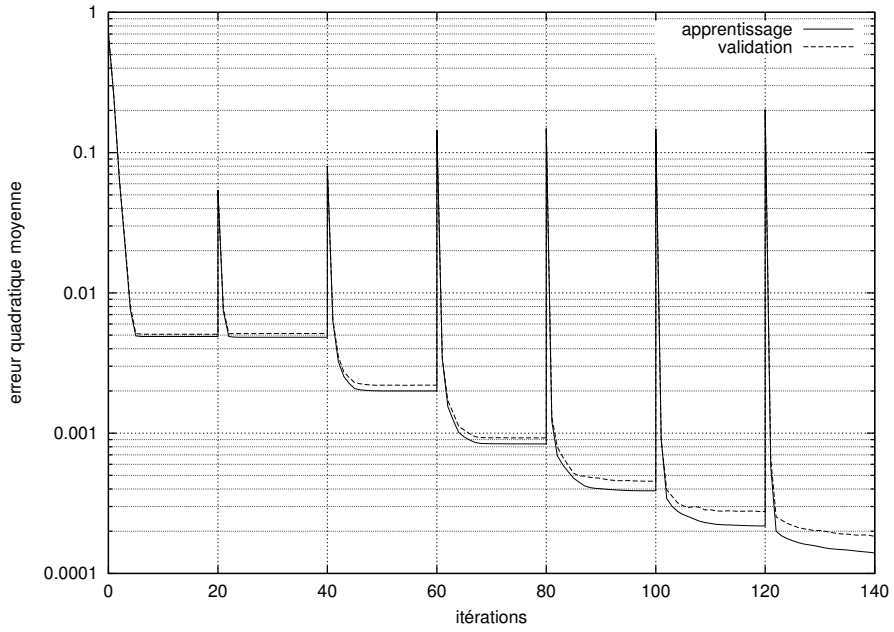

Figure 3.21. Apprentissage d'un RPN dans un système OFDM à 4 porteuses avec un amplificateur SSPA, une modulation MAQ16, un recul de 0 dB, une base de 16384 éléments sans bruit et un algorithme de Levenberg Marquardt

L'apprentissage a été arrêté à l'ordre 7 car pour des ordres supérieurs on ne constate plus de diminution de l'erreur quadratique moyenne. On constate bien sur cette courbe que le modèle réalisé par le réseau RPN est affiné à chaque fois que l'on ajoute un réseau pi-sigma. Le réseau est ensuite simulé dans la chaîne OFDM afin de mesurer le taux d'erreur binaire en sortie :

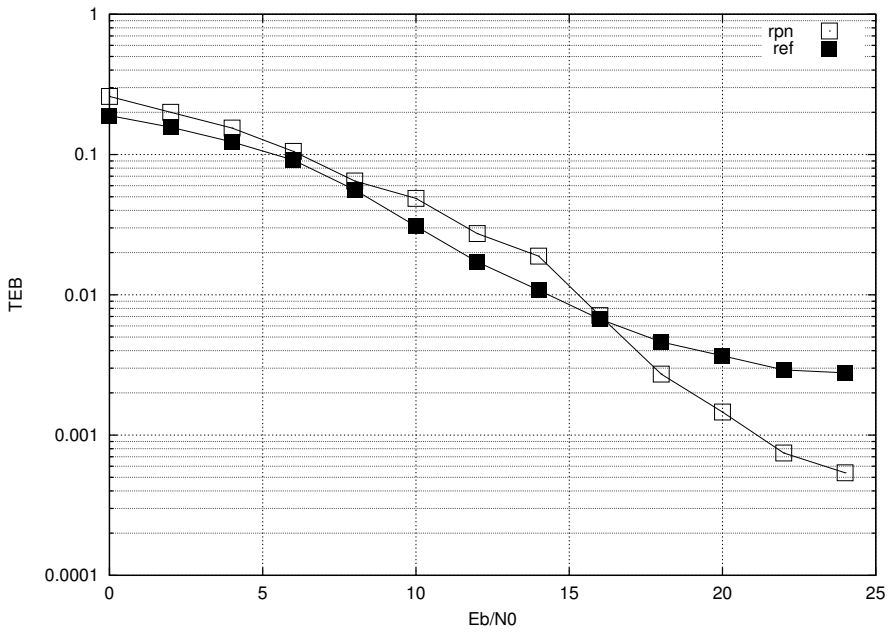

Figure 3.22. Taux d'erreur binaire d'un système OFDM à 4 porteuses avec une modulation MAQ16, un amplificateur SSPA, un recul de 0 dB et correcteur RPN (apprentissage sans bruit)

La courbe 'rpn' correspond à celle obtenue avec le correcteur RPN, et la courbe 'ref' est celle de référence, sans correcteur, mais toujours avec la non-linéarité. Le correcteur avec le réseau RPN n'est efficace qu'au dessus d'un certain rapport signal sur bruit (16 dB environ). Nous avions évoqué dans la section 2.2 p. 73 qu'il était possible qu'il soit nécessaire d'ajouter du bruit lors de l'apprentissage pour augmenter l'immunité au bruit du réseau de neurones en utilisation après son apprentissage, c'est ce que nous allons illustrer maintenant.

4.2. Apprentissage et résultats du réseau RPN sur une base avec bruit

Nous avons donc réalisé plusieurs apprentissages avec des rapports signal sur bruit différents, puis simulé tous les réseaux afin d'extraire le taux d'erreur binaire obtenu dans le système OFDM. Tous les autres paramètres sont identiques à ceux du dernier réseau RPN : 4 porteuses, modulation MAQ16, un amplificateur non linéaire SSPA avec le paramètre p=2, réseau RPN d'ordre 7, algorithme de Levenberg Marquardt, et 16384 exemples dans la base. Les courbes d'apprentissage des réseaux sont très similaires à celles de la figure 3.21, sauf l'erreur quadratique moyenne qui est plus élevée, à cause du bruit sur la base. La courbe figure 3.23 montre les taux d'erreur binaire avec tous ces réseaux :

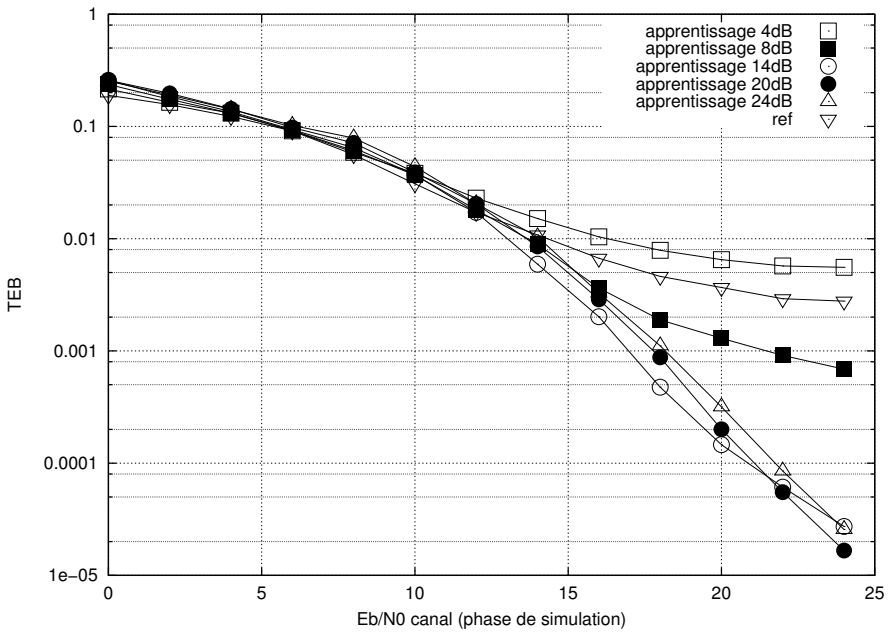

Figure 3.23. Taux d'erreur binaire d'un système OFDM à 4 porteuses avec une modulation MAQ16, un amplificateur SSPA, un recul de 0 dB et correcteur RPN (apprentissage avec bruit)

La courbe 'ref' est toujours la référence, et pour toutes les autres courbes, le rapport signal sur bruit qui a été ajouté à la base d'apprentissage est indiqué. On remarque que ce bruit ajouté à la base d'apprentissage joue un très grand rôle dans les performances du correcteur. Pour un rapport E_b/N_0 trop faible, comme 4 dB, le niveau de bruit est tellement élevé que l'algorithme d'optimisation ne permet pas de trouver un réseau de neurones avec de bonnes performances. En diminuant la puissance du bruit, on améliore les performances du correcteur, et le taux d'erreur binaire est bien réduit. On remarque que le réseau ayant appris avec $E_b/N_0 = 14\mathrm{dB}$ a les meilleurs performances quand il est simulé avec un canal présentant un E_b/N_0 inférieur à 21 dB. Mais ensuite, un réseau ayant appris avec moins de bruit (un taux de 20 dB) a de meilleures performances. L'interprétation de ces résultats étant difficile, une autre courbe a été tracée figure 3.24 dans le but de trouver le meilleur rapport signal sur bruit à utiliser pour créer la base d'apprentissage. En ordonnée nous avons toujours le taux d'erreur binaire, et en abscisse cette fois il s'agit du rapport E_b/N_0 de la base d'apprentissage. Les courbes sont tracées à E_b/N_0 de en phase d'exploitation fixe.

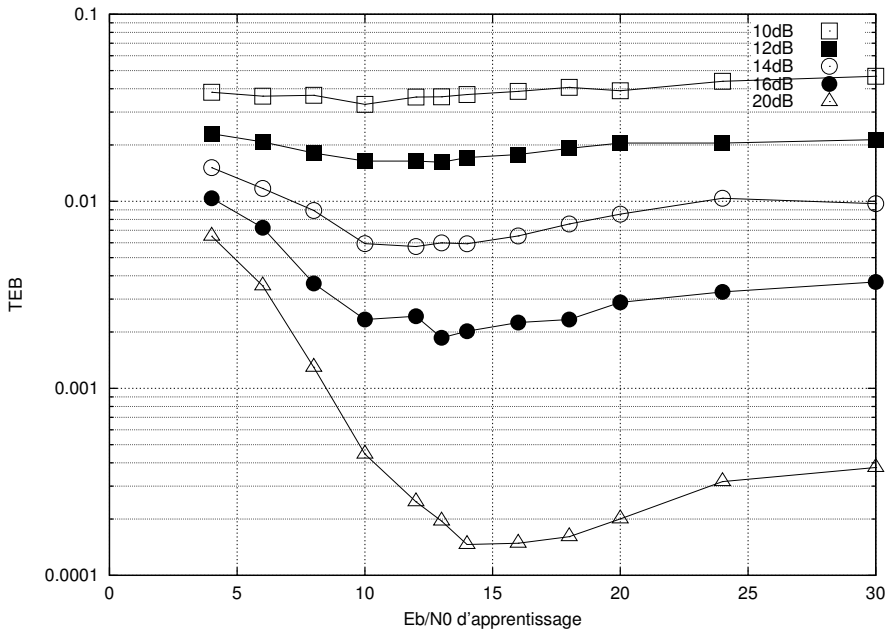

Figure 3.24. Taux d'erreur binaire d'un système OFDM à 4 porteuses (recul de 0 dB, amplificateur SSPA, modulation MAQ16) avec correcteur RPN (simulation à Eb/N0 fixe sur chaque courbe, apprentissage avec différents Eb/N0)

On remarque que quel que soit le rapport signal sur bruit en simulation, ce sont les bases avec un rapport signal sur bruit situé entre environ 10 et 20 dB qui permettent d'obtenir les meilleures performances. Pour un E_b/N_0 en exploitation de 10 dB, le minimum de la courbe d'erreur correspond à une base d'apprentissage avec un E_b/N_0 de 10 dB, et lorsque l'on augmente le rapport signal sur bruit en exploitation, ce minimum augmente également jusqu'à atteindre 15 dB pour un E_b/N_0 en exploitation de 20 dB. Nous avons choisi $E_b/N_0 = 13\mathrm{dB}$ pour la base d'apprentissage, ce qui permet d'obtenir un taux d'erreur binaire faible dans la plupart des cas simulés et représente un bon compromis. C'est donc ce taux qui sert de référence et qui sera toujours utilisé dans la constitution de bases d'apprentissage pour ce correcteur RPN.

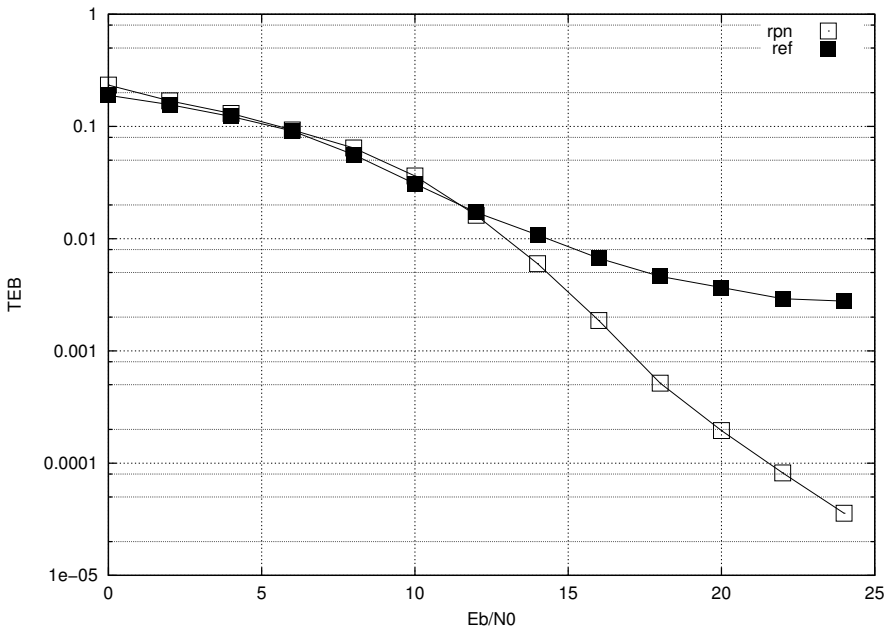

Figure 3.25. Taux d'erreur binaire d'un système OFDM à 4 porteuses avec une modulation MAQ16, un amplificateur SSPA, un recul de 0 dB et correcteur RPN (apprentissage avec E_b/N_0=13 dB)

La figure 3.25 présente donc les résultats avec le réseau RPN et la base d'apprentissage choisis. Pour un taux d'erreur binaire de 10^{-2} on mesure un gain apporté par le correcteur d'environ 1,5 dB. Pour un taux d'erreur binaire de $5 \cdot 10^{-3}$, ce gain est d'environ 4 dB. Pour un rapport E_b/N_0 de 20 dB le taux d'erreur binaire est divisé par 19 grâce au réseau de neurones. Ces résultats ont été publiés dans [TERT02] et [TERT03].

4.3. Autres non-linéarités

Un des avantages des réseaux de neurones est leur capacité à s'adapter à différents problèmes. Nous avons donc entraîné puis simulé le même réseau RPN dans un système OFDM avec une non-linéarité différente. Dans un premier temps nous avons pris un amplificateur de type SSPA avec le paramètre p=3 (ce modèle sera noté SSPA3 dans le reste de ce mémoire), et comme on peut le constater après apprentissage le correcteur parvient toujours à corriger les non-linéarités de manière significative :

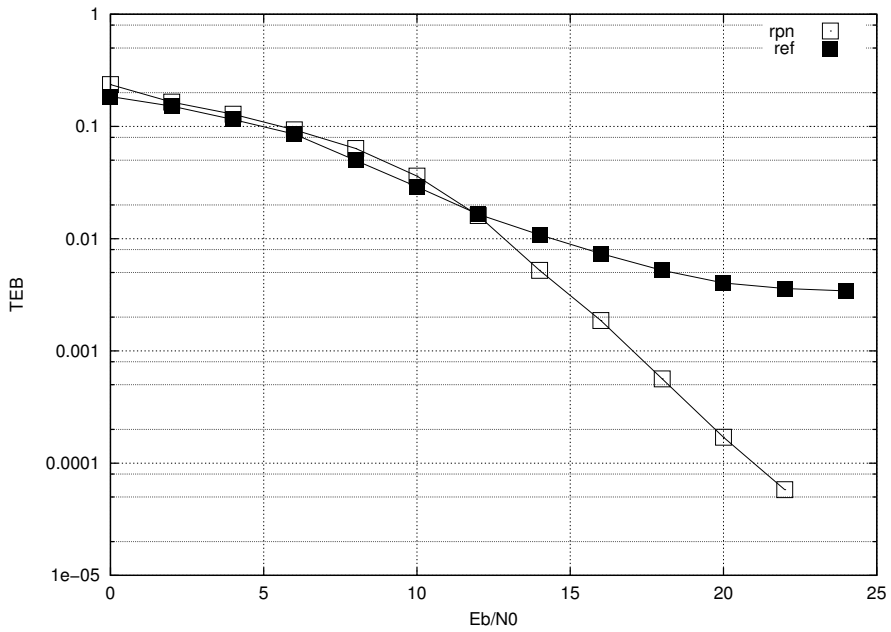

Figure 3.26. Taux d'erreur binaire d'un système OFDM à 4 porteuses avec une modulation MAQ16, un amplificateur SSPA3, un recul de 0 dB et correcteur RPN, (apprentissage avec Eb/N0=13 dB)

La courbe est comparable à celle obtenue avec le modèle SSPA prenant p=2 (figure 3.25), nous constatons les mêmes gains de 1,5 dB et 4 dB pour les taux d'erreur binaire respectifs de 10^{-2} et 10^{-3}. Pour un rapport E_b/N_0 de 20 dB le taux d'erreur binaire est divisé par 24 avec le correcteur. Enfin nous avons pris un limiteur, et dans ce cas également le correcteur est efficace :

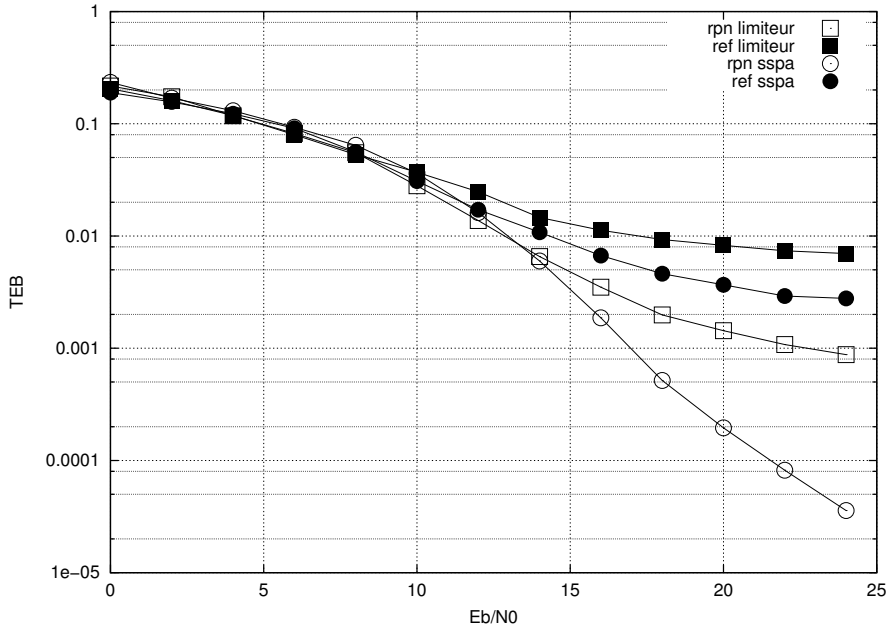

Figure 3.27. Taux d'erreur binaire d'un système OFDM à 4 porteuses avec correcteur RPN, limiteur avec un recul de 0 dB, modulation MAQ16, apprentissage avec E_b/N_0=13 dB, et comparaison avec les résultats obtenus avec le modèle SSPA p=2

La courbe de référence (sans correcteur mais avec le limiteur) est appelée 'ref limiteur', et celle obtenue avec le correcteur RPN est appelée 'rpn limiteur'. Le gain apporté par le réseau de neurones est de presque 5 dB pour un taux d'erreur binaire de 10^{-2} et de plus de 10 dB pour un taux d'erreur binaire de $5 \cdot 10^{-3}$. Pour un rapport E_b/N_0 de 20 dB le taux d'erreur binaire est divisé par 7 avec le correcteur. Ainsi avec un nouvel apprentissage, le correcteur à RPN est capable de s'adapter à des non-linéarités différentes. En pratique ceci peut être intéressant dans une application mobile, dans laquelle le récepteur peut se déplacer, et nécessiter une connexion avec un nouvel émetteur lors d'un changement de zone. Si le nouvel émetteur possède un amplificateur différent de l'ancien, il suffira au récepteur d'effectuer un nouvel apprentissage pour s'adapter, à condition que le temps et la puissance de calcul requis par cette apprentissage soient compatibles avec l'application voulue.

Les gains apportés par le correcteur à taux d'erreur binaire fixé semblent indiquer que le réseau est beaucoup plus efficace dans le cas du limiteur que les autres cas, ce qui n'est pas forcément évident en regardant les courbes. Dans la figure 3.27 nous avons ajouté les résultats présentés figure 3.25 dans le cas d'une non linéarité de type SSPA avec p=2 (courbes 'ref sspa' et 'rpn sspa'). On peut remarquer que dans le cas du limiteur la courbe du système sans correction présente une sorte de plateau à un taux d'erreur binaire plus élevé que dans celui de l'amplificateur SSPA, ce qui augmente les gains mesurés à taux d'erreur binaire fixe. Ainsi pour comparer les performances du correcteur basé sur un réseau de neurones dans des systèmes OFDM ayant des caractéristiques différentes, la mesure du gain en taux d'erreur binaire à rapport signal sur bruit fixe est plus fiable. Les mesures présentées ont été faites à un rapport E_b/N_0 de 20 dB, et le taux d'erreur binaire a été divisé par 7 dans le cas du limiteur et 19 dans le cas de l'amplificateur SSPA avec p=2, ce qui est plus conforme à ce que l'on constate graphiquement.

4.4. Augmentation du nombre de porteuses

Après avoir obtenu des gains significatifs avec le réseau RPN sur un système à 4 porteuses dans les conditions évoquées précédemment, nous avons fait des essais avec 8 et 16 porteuses. Considérons tout d'abord le cas avec 8 porteuses : nous avons construit une base d'apprentissage de 32768 symboles OFDM, avec un rapport signal sur bruit de 13 dB et entraîné le un RPN par un algorithme de Levenberg-Marquardt. La figure 3.28 montre les évolutions des erreurs d'apprentissage et de validation au fur et à mesure des itérations :

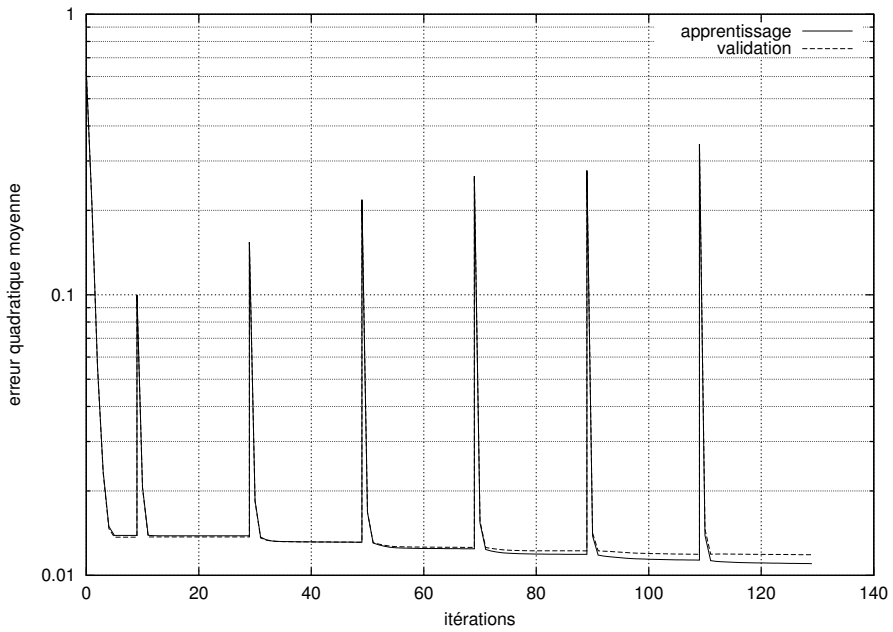

Figure 3.28. Apprentissage d'un RPN appliqué à un système OFDM à 4 porteuses avec un amplificateur SSPA, une modulation MAQ16, un recul de 0 dB et une base de 32768 éléments avec bruit (E_b/N_0=13 dB)

Comme pour l'apprentissage avec 4 porteuses (figure 3.21), l'erreur quadratique moyenne ne diminue pas lorsque l'on ajoute des réseaux pi-sigma d'ordre supérieur à 7, et nous nous sommes donc arrêtés à cet ordre. Le réseau RPN ainsi entraîné est ensuite simulé dans un système OFDM :

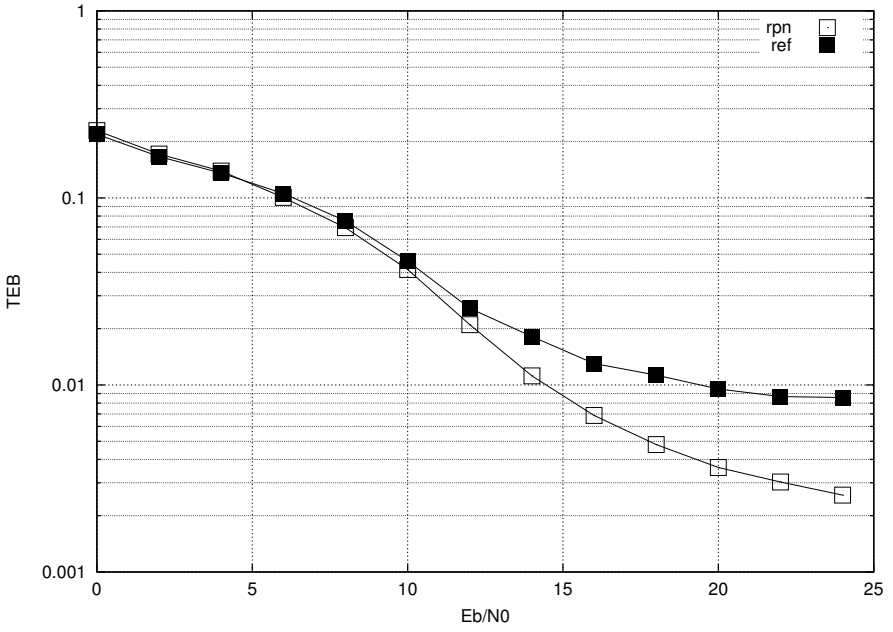

Figure 3.29. Taux d'erreur binaire sur un système OFDM à 8 porteuses avec une modulation MAQ16, un amplificateur SSPA, un recul de 0 dB et correcteur RPN (apprentissage avec E_b/N_0=13 dB)

On constate un gain de 5 dB pour un taux d'erreur binaire de $5 \cdot 10^{-2}$, mais les deux courbes sont tout de même plus proches que lorsqu'il n'y avait que 4 porteuses, comme on peut le voir sur la

figure 3.30 qui présente les deux résultats montré figures 3.25 et 3.29. A titre de comparaison la courbe théorique sans non-linéarités est également présente.

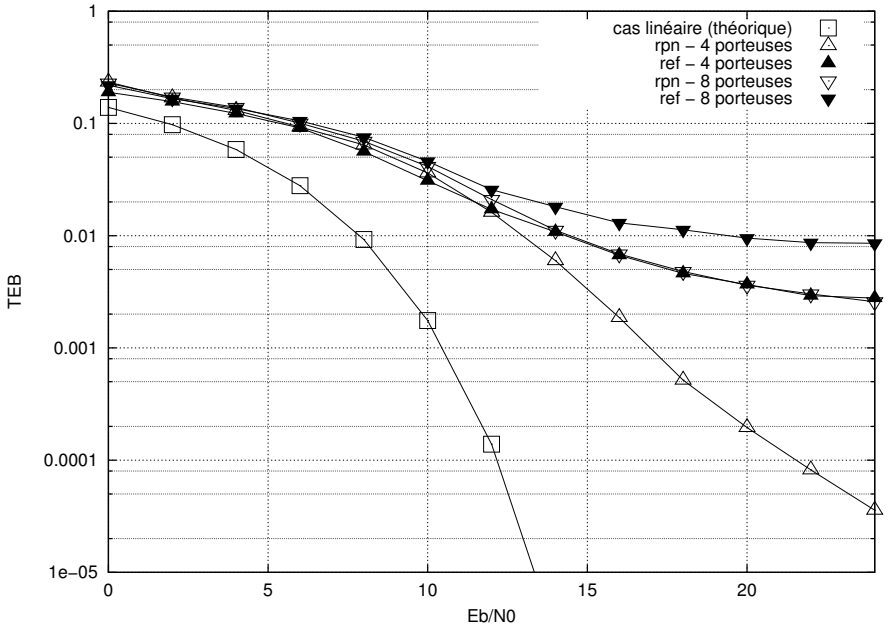

Figure 3.30. Comparaison des résultats obtenus avec le correcteur RPN simulé dans des systèmes OFDM à 4 et 8 porteuses, avec un amplificateur SSPA, un recul de 0 dB et une modulation MAQ16

Pour un rapport E_b/N_0 de 20 dB le taux d'erreur binaire est divisé par 3 seulement, ce qui est effectivement bien moins que le rapport 19 mesuré dans le système avec 4 porteuses. Pour le système avec 16 porteuses nous avons à nouveau doublé la taille de la base, portée donc à 65536 éléments. L'apprentissage du réseau est plus difficile :

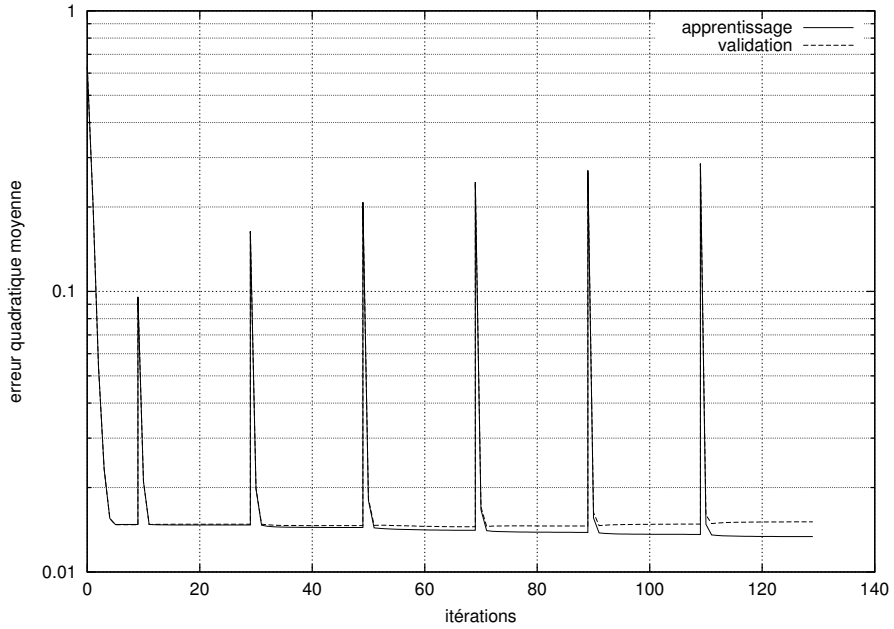

Figure 3.31. Apprentissage d'un RPN sur un système OFDM à 16 porteuses avec un amplificateur SSPA, une modulation MAQ16, un recul de 0 dB et une base de 65536 éléments avec bruit (E_b/N_0=13 dB)

On constate en effet que l'erreur quadratique moyenne sur la base d'apprentissage ne diminue pas lors de chaque ajout d'un réseau pi-sigma, et celle de validation augmente même à partir de l'ordre 4. Les algorithmes d'apprentissage (descente de gradient et Levenberg-Marquardt) ne parviennent pas à affiner le modèle et la performance à la fin de l'apprentissage reste la même que celle obtenue avec un réseau pi-sigma d'ordre 1, c'est à dire d'un système linéaire. On peut donc supposer que le correcteur ainsi obtenu ne pourra pas diminuer le taux d'erreur binaire par rapport à un système classique. Ceci est confirmé par la simulation :

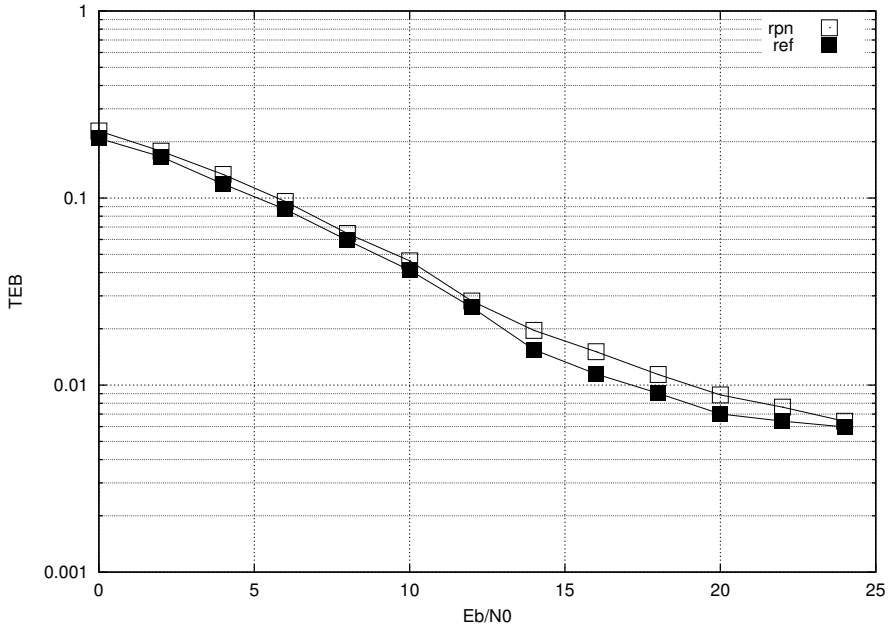

Figure 3.32. Taux d'erreur binaire avec le correcteur RPN appliqué à un système OFDM à 16 porteuses avec une modulation MAQ16, un amplificateur SSPA, un recul de 0 dB et correcteur RPN (apprentissage avec E_b/N_0=13 dB)

Le taux d'erreur est plus élevé avec le correcteur RPN (courbe 'rpn') que sans correcteur (courbe 'ref'). Nous avons utilisé diverses méthodes pour améliorer l'apprentissage : augmenter la taille de la base d'apprentissage, réduire le bruit sur la base, prendre une modulation plus simple (MAQ4) et revenir à une descente de gradient adaptative. Aucune de ces méthodes n'a permis d'améliorer les performances du correcteur. Comme pour les PMC, dans ce cas là l'algorithme d'apprentissage ne permet pas de converger vers un réseau ayant les caractéristiques recherchées. Nous constatons le même problème lorsque le nombre de porteuses est supérieur à 16, des tests ayant également été réalisés avec 32 et 48 porteuses.

Plusieurs voies de recherches peuvent être explorées pour tenter de résoudre le problème. Un travail sur l'algorithme d'apprentissage lui-même, avec par exemple une meilleure utilisation des symétries, ou une optimisation de la constitution de la base. Une autre voie qui pourrait être prometteuse est la famille de réseaux de neurones SVM (Support Vector Machines) [VAPN98]. Un réseau SVM peut avoir de bonnes performances dans un espace d'entrées à grand nombre de dimensions, et ne présente pas les problèmes d'apprentissage des PMC et autres réseaux d'ordre supérieur dus à la présence de minimums locaux. Le principe d'un réseau SVM est de s'appuyer sur quelques exemples de la base, les vecteurs de support, pour calculer la fonction de sortie. Un algorithme permet de sélectionner les exemples les plus significatifs, situés généralement à la frontière entre deux zones différentes de l'espace d'entrée. Ces deux axes n'ont pas été étudiés.

Une autre méthode qui peut être utilisée est de faire un traitement des données avant de les transmettre au réseau de neurones. C'est cette démarche qui a été suivie, en le plaçant dans le domaine temporel, comme nous le verrons dans le chapitre suivant.

Conclusion

Dans ce chapitre nous avons présenté une première méthode permettant de réduire les effets des non-linéarités dans un système OFDM, à l'aide d'un réseau de neurones travaillant dans le domaine fréquentiel. A partir des symboles reçus sur toutes les porteuses à un instant donné, le réseau de neurones apprend à retrouver les symboles émis, et donc à compenser les effets de la non-linéarité de l'amplificateur.

Parmis tous les réseaux de neurones testés, celui permettant d'obtenir les meilleurs performances est le réseau RPN, un réseau d'ordre supérieur. Le correcteur basé sur un RPN permet de réduire le taux d'erreur binaire sur la transmission OFDM avec 4 et 8 porteuses avec un amplificateur SSPA ou un limiteur, avec un recul de 0 dB et une modulation MAQ16. Avec 4 porteuses le taux d'erreur binaire est divisé par 19 avec un amplificateur SSPA à p=2 et par 7 avec un limiteur, pour un rapport signal sur bruit de 20 dB. On constate de plus qu'il existe un rapport signal sur bruit optimal lors de la constitution de la base d'apprentissage pour obtenir un correcteur avec les meilleures performances en exploitation. Le réseau a tout d'abord été entraîné avec une descente de gradient, mais l'algorithme de Levenberg Marquardt a permis de diminuer de manière significative la durée de l'apprentissage, qui a été réduite de 1 heure et 50 minutes à 14 minutes.

Par contre lorsque le nombre de porteuses est supérieur ou égal à 16, l'architecture que nous avons retenue et l'algorithme d'apprentissage que nous avons utilisé ne permettent plus d'obtenir un résultat satisfaisant.

Finalement, plutôt que de chercher une architecture qui permettrait d'augmenter le nombre de porteuses nous avons essayé de réduire la complexité du problème en recherchant une solution non plus dans le domaine fréquentiel, mais dans le domaine temporel. Cette méthode et les résultats obtenus sont présentés dans le chapitre suivant.

Chapitre 4. Compensation des non-linéarités dans le domaine temporel

Introduction

Dans le chapitre précédent, la méthode pour compenser les non-linéarités de l'amplificateur dans le domaine fréquentiel a été présentée. Cette méthode a donné de bons résultats pour un faible nombre de porteuses, allant jusqu'à 8. Mais au delà, des problèmes de convergence apparaissent. Nous avons donc cherché un moyen de réduire la complexité du correcteur, en le plaçant dans le domaine temporel.

Des réseaux de neurones ont déjà été étudiés pour de l'égalisation non linéaire dans le domaine temporel, mais uniquement dans le cadre de modulations monoporteuses. Le but fixé est donc tout d'abord d'adapter l'une de ces méthodes aux multiporteuses, puis ensuite de sélectionner parmi plusieurs architectures neuronales celle qui fournit les meilleures performances. Dans ce chapitre nous allons tout d'abord présenter la méthode retenue, puis présenter des résultats de simulation.

1. Principe

Dans un système monoporteuse, l'égalisation du canal est généralement réalisée dans le domaine temporel, à l'aide d'un filtre adaptatif. Pour compenser de plus les effets non linéaires de l'amplificateur, il est possible d'utiliser des filtres de Volterra [BIGL84], qui sont en quelque sorte une extension du filtrage adaptatif au cas non-linéaire. Dans ce cas le filtre réalisera à la fois l'égalisation du canal et celle de la non-linéarité de l'amplificateur. Les réseaux de neurones ont également été étudiés dans ce contexte et des égaliseurs basés sur des PMC [ERDO01] et des GRBF [CHAN98] ont été proposés, comme indiqué dans la section 1.1 p. 65. Revenons sur ces travaux.

La figure 4.1 présente le principe d'un égaliseur non linéaire réalisé avec un perceptron à une couche cachée. Dans ce système le réseau de neurones effectue également l'égalisation du canal. En effet en plus du symbole reçu à un instant t, un certain nombre de symboles précédents sont fournis au réseau. Comme le réseau n'utilise que des signaux réels, les symboles complexes sont séparés en parties réelle (I) et imaginaire (Q).

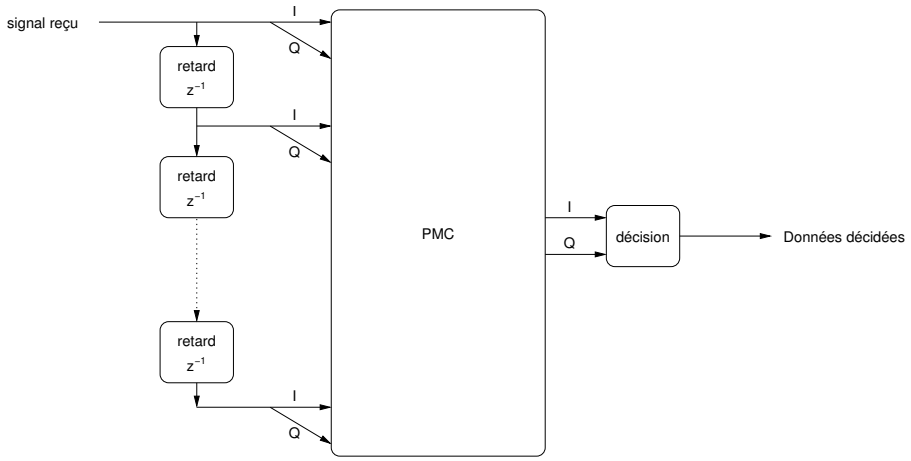

Figure 4.1. Principe d'un égaliseur non linéaire réalisé entièrement par un perceptron multicouches

Le nombre d'entrées du réseau de neurones est donc le double de la longueur de la réponse impulsionnelle du filtre réalisé, c'est à dire de celle de la réponse du canal. Le défaut d'une telle approche est que si le canal possède une réponse impulsionnelle longue, ce qui est le cas généralement des canaux multitrajets, il sera nécessaire de donner un grand nombre d'entrées au réseau, ce qui veut dire un grand nombre de poids à ajuster et donc un apprentissage plus long. C'est pour cela qu'une autre méthode a été proposée, nommée égaliseur neuronal hybride [BOUC99].

Dans un égaliseur hybride, l'égalisation du canal est effectuée par un égaliseur classique, et celle de la non-linéarité de l'amplificateur par un PMC ou un GRBF. La figure 4.2 présente le schéma de principe d'un PMC associé à un égaliseur de canal de type DFE (Decision Feedback Equaliser). Cette fois, un filtre linéaire est placé avant le réseau de neurones, et ce dernier ne possède ainsi que deux entrées.

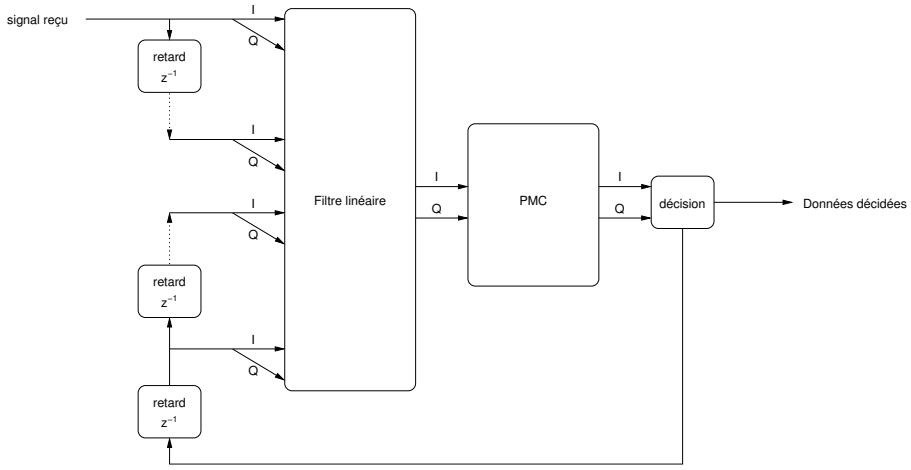

Figure 4.2. Principe d'un égaliseur non linéaire hybride combinant un PMC et un filtre de type DFE

C'est ce dernier principe que nous allons appliquer à la modulation multiporteuses, en combinant, comme dans la solution présentée dans le chapitre précédent, l'égalisation linéaire dans le domaine fréquentiel spécifique la modulation OFDM (voir section 2.3 p. 23), et un réseau de neurones dans le domaine temporel qui effectue la partie non linéaire de l'égalisation. Comme

rappelé en section 1.1 p. 65 et figure 3.1, l'ordre dans lequel sont placés les différents éléments est important. La compensation des non-linéarités doit être placée après l'égalisation du canal. Comme cette dernière est réalisée dans le domaine fréquentiel, il va être nécessaire de projeter le signal dans le domaine temporel pour effectuer la compensation de la non-linéarité, puis à nouveau le projeter dans le domaine fréquentiel afin de retrouver les symboles OFDM. La figure 4.3 présente un schéma d'un tel récepteur. TFDI signifie Transformée de Fourier Discrète Inverse.

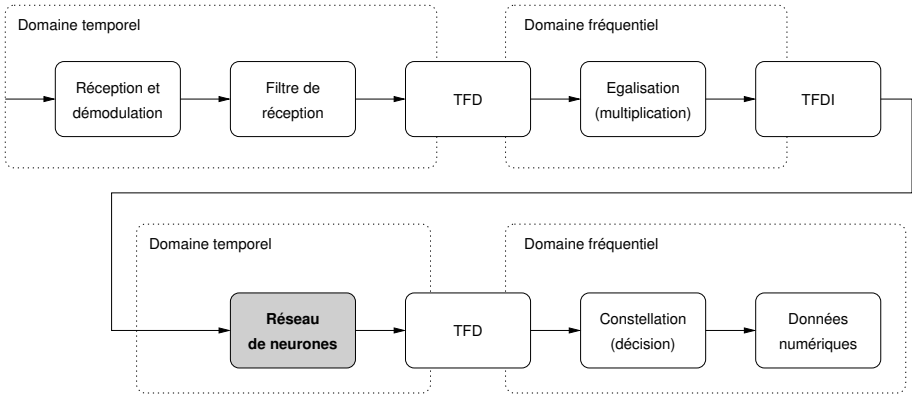

Figure 4.3. Schéma d'un récepteur OFDM avec une compensation des non-linéarités dans le domaine temporel

Le système est plus complexe que dans la méthode précédente, à cause des deux transformées de Fourier que l'on doit ajouter pour effectuer les changements de domaines. Par contre le réseau de neurones lui-même sera plus simple, puisque ses espaces d'entrée et de sortie n'ont que deux dimensions. Enfin un tel système est indépendant du nombre de porteuses et des modulations choisies pour la transmission numérique. En effet dans le cas fréquentiel, le nombre d'entrées et de sorties du réseau est égal au double du nombre de porteuses, et la fonction d'activation à choisir pour la couche de sortie dépend de la modulation. Par contre dans le cas temporel, ces paramètres n'influent pas sur l'architecture du réseau.

Il y a deux différences entre le correcteur temporel que l'on désire réaliser ici et les égaliseurs neuronaux hybrides cités précédemment. Tout d'abord l'égalisation linéaire est ici effectuée par une multiplication dans le domaine fréquentiel, adaptée à la modulation OFDM. Ensuite, dans le cas monoporteuse, le réseau de neurones réalise une classification, puisqu'il doit reconnaître les symboles d'une constellation. Dans le cas de l'OFDM, les informations sont dans le domaine fréquentiel, et le signal temporel peut prendre un nombre de valeurs différentes bien plus large. Le réseau de neurones doit faire une approximation de fonction sur un intervalle, et non plus une classification en un petit nombre de cas. Ainsi les architectures neuronales qui étaient adaptées au cas monoporteuse ne le sont pas obligatoirement en multiporteuses, et réciproquement.

2. Égalisation avec le modèle de l'amplificateur

2.1. Application directe du modèle inverse

Comme le signal OFDM temporel possède un facteur de crête élevé, il est constitué de pics qui peuvent être d'amplitude bien plus élevée que le seuil de saturation de l'amplificateur. Les effets des non-linéarités (harmoniques et intermodulations) peuvent donc être plus importants qu'avec un signal monoporteuse, à recul équivalent. Or un certain nombre d'intermodulations et toutes les harmoniques se situent à des fréquences qui sont en dehors de la bande utilisée pour la communication, et donc fortement atténuées par le filtre de réception. Dans le cas général ceci est bénéfique, car les intermodulations sont des perturbations du signal. Mais dans cette application, ceci veut dire que le signal après le filtre de réception n'est plus le même que celui présent directement à la sortie de l'amplificateur. Donc même si l'on place dans le récepteur un système qui inverse parfaitement la non-linéarité de l'amplificateur, il subsistera des perturbations du signal et éventuellement des erreurs. Avant de tester quelques correcteurs à base de réseaux de neurones, nous avons voulu mesurer les performances d'un tel correcteur théorique, afin de vérifier que le filtrage des harmoniques et intermodulations ne nuit pas de manière significative à la transmission.

Le modèle d'amplificateur que nous avons utilisé pour nos simulations est présenté dans le paragraphe p. 66, équations (3.1) et (3.2). Un calcul simple nous permet de trouver la réciproque de ce modèle, la caractéristique du modèle inverse :

$$\begin{aligned} & s(t) = s_0(t) \cdot f(|s_0(t)|) \\ & \text{où } f(r) = \frac{1}{\nu(1-(\frac{r}{A_0})^{2p})^{\frac{1}{2p}}} \end{aligned} \tag{4.1}$$

où ν est le gain de l'amplificateur et A_0 l'amplitude de saturation en sortie. Nous avons simulé ce correcteur dans un système OFDM avec 4 porteuses, une modulation MAQ16, un recul de 0 dB et $p = 2$ dans le modèle SSPA. Le taux d'erreur binaire d'un tel système en fonction du rapport signal sur bruit E_b/N_0 est présenté figure 4.4, par la courbe 'sspa'. La courbe 'ref' présente ce même taux d'erreur binaire dans un système sans correcteur mais toujours avec la non-linéarité.

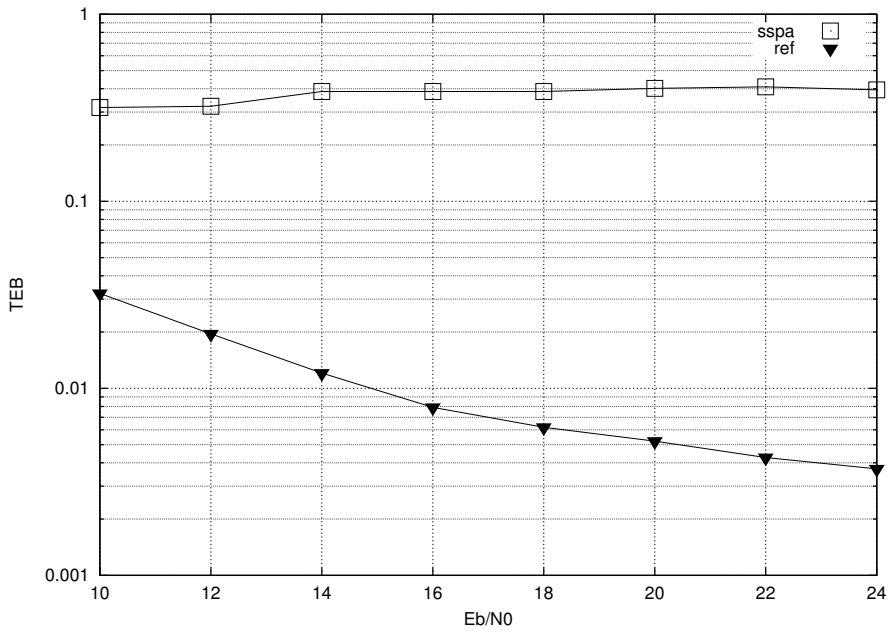

Figure 4.4. Taux d'erreur binaire d'un système OFDM à 4 porteuses, un recul de 0 dB, une modulation MAQ16 et correcteur SSPA inversé

On constate donc qu'un tel égaliseur ne fait qu'ajouter des erreurs à la transmission : pour un rapport signal sur bruit de 24 dB le taux d'erreur binaire est toujours de 0,3, et le système sans correcteur a des performances bien meilleures. Pour expliquer ce phénomène il faut étudier un échantillon du signal OFDM temporel. La figure 4.5 présente le module d'un tel échantillon, avant amplification, après amplification non linéaire, et après correction par le modèle SSPA inversé (avec un bruit de $E_b/N_0 = 10dB$ dans le canal).

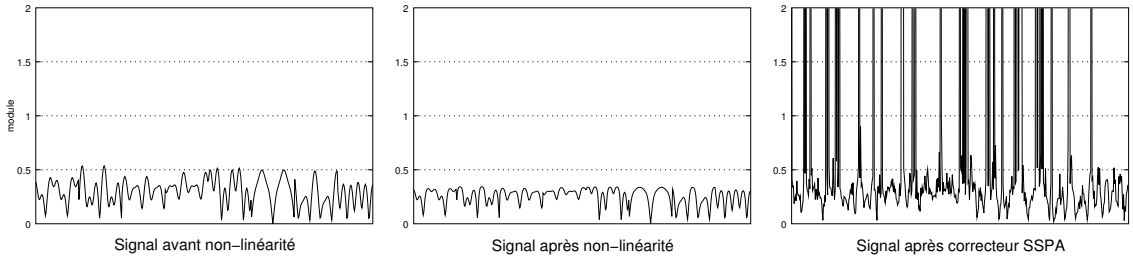

Figure 4.5. Module d'un signal OFDM temporel à différents stades de la chaîne de transmission

Comme on peut le constater le signal se retrouve "aplati" après passage dans l'amplificateur non linéaire. Par contre après le passage dans le correcteur, on remarque la présence de pics d'amplitudes très importantes. Ces pics sont créés par le correcteur, qui amplifie le bruit du canal additif gaussien. La figure 4.6 montre l'amplitude de la sortie du correcteur en fonction de celle de l'entrée. La réponse du correcteur possède une pente très importante près de la saturation, et le bruit additif est très fortement amplifié lorsque le signal OFDM temporel se trouve dans cette zone.

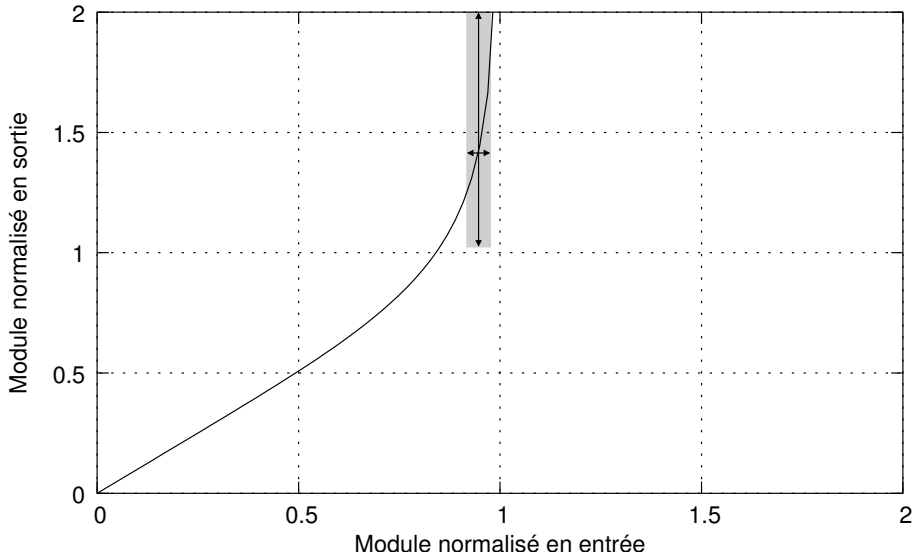

Figure 4.6. Réponse en module du correcteur SSPA inversé

L'entrée est normalisée de telle manière que l'amplitude 1 corresponde au seuil de saturation du modèle SSPA (A_0) et la sortie de telle manière que le gain du correcteur soit de 1 dans la zone linéaire.

Un moyen envisageable pour pallier ce problème est de limiter l'amplitude du signal en sortie du correcteur, en ajoutant une marge de saturation.

2.2. Ajout d'une marge de saturation

Pour limiter l'amplitude du signal de sortie, lorsque le module du signal à l'entrée du correcteur est supérieur à un certain seuil, il est ramené à ce seuil. Le rapport entre l'amplitude de saturation et le seuil sera appelé marge de saturation, et exprimé en dB. Afin de trouver la marge adéquate, le système OFDM a été simulé avec différentes marges et un rapport signal sur bruit $E_b/N_0 = 20\mathrm{dB}$. La figure 4.7 montre le taux d'erreur binaire en fonction de la marge :

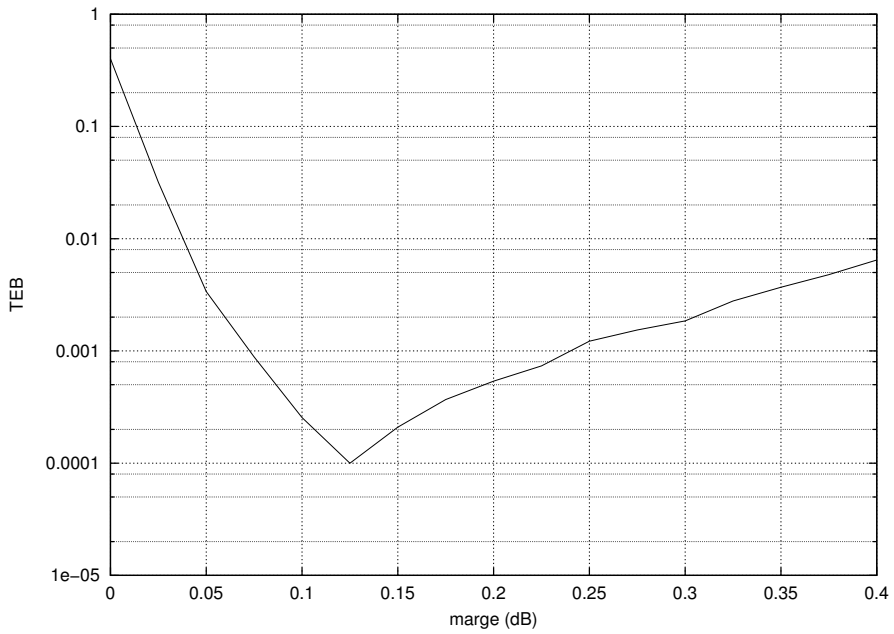

Figure 4.7. Taux d'erreur binaire dans une chaîne OFDM à 4 porteuses, modulation MAQ16, correcteur SSPA inversé, recul de 0 dB, en fonction de la marge de saturation

Une marge de saturation de 0.125 dB est donc celle qui permet le taux d'erreur binaire le plus faible dans ce cas. La figure 4.8 montre le taux d'erreur binaire en fonction du rapport signal sur bruit avec cette marge de saturation :

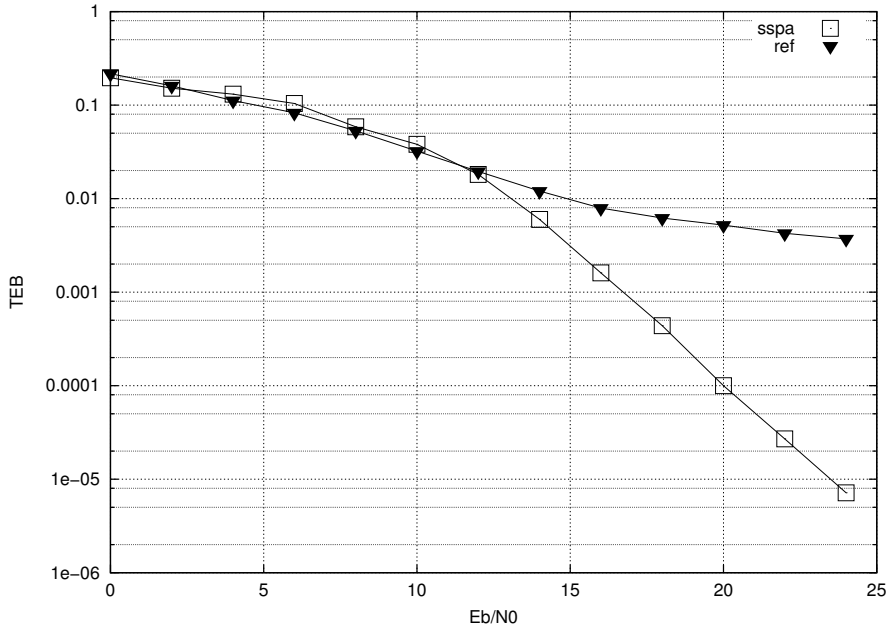

Figure 4.8. Taux d'erreur binaire dans une chaîne OFDM à 4 porteuses, modulation MAQ16, correcteur SSPA inversé avec marge de saturation de 0.125 dB et recul de 0 dB

Cette fois on constate une réelle amélioration de la qualité de la transmission, le correcteur SSPA inversé apporte un gain de 2 dB pour un taux d'erreur binaire de 10^{-2}. En appliquant la même méthode à un système OFDM avec 48 porteuses (ce qui correspond à la modulation

utilisée dans les réseaux HiperLan/2), on conclue que la meilleure marge vis-à-vis du taux d'erreur binaire est de 0.15 dB, et l'on constate des performances de même nature, ce qui confirme que ce système est peu gêné par une augmentation du nombre de porteuses :

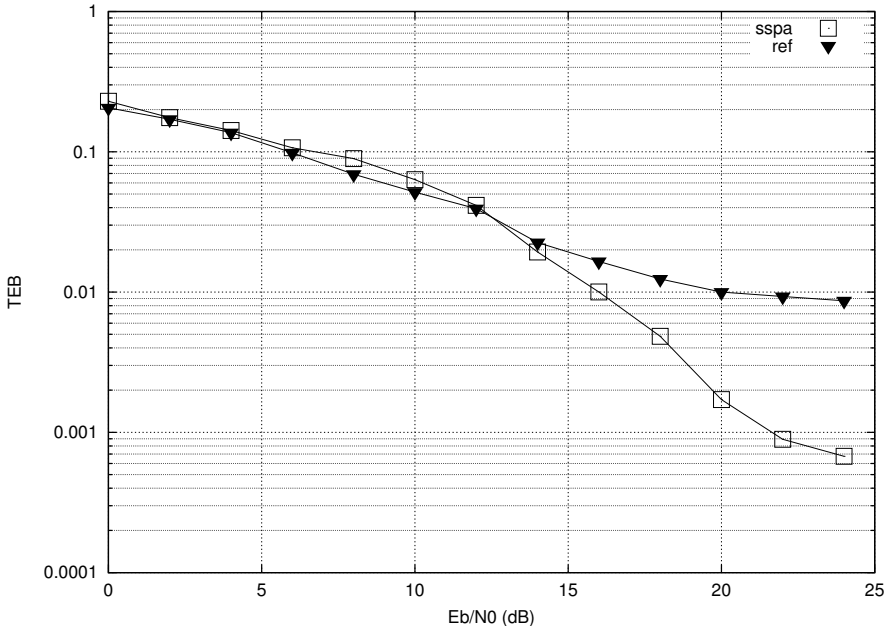

Figure 4.9. Taux d'erreur binaire dans une chaîne OFDM à 48 porteuses, modulation MAQ16, correcteur SSPA inversé avec marge de saturation de 0.15 dB et recul de 0 dB

En augmentant encore le nombre de porteuses, on détermine par simulation qu'il faut augmenter encore légèrement la marge de saturation, qui est par exemple de 0.2 dB pour 512 porteuses. Mais cette marge reste du même ordre de grandeur dans tous les cas simulés.

On constate donc que ce correcteur basé sur le modèle SSPA permet de réduire le nombre d'erreurs dues aux non-linéarités. Cependant ce système peut difficilement être mis en pratique, pour plusieurs raisons. Tout d'abord, il implique de connaître parfaitement le modèle de l'amplificateur non linéaire. Si la réponse de celui-ci ne correspond pas au modèle qui a été utilisé pour déterminer la réponse du correcteur, il est probable qu'il ne sera plus efficace. De plus le signal ne doit pas être amplifié ou atténué : le signal à l'entrée du correcteur inverse doit avoir exactement la même amplitude que celui qui sortait de l'amplificateur, sinon l'inversion ne sera plus valide (cependant ce second point pourrait être résolu à l'aide d'un gain adaptatif). Enfin, rien ne prouve que l'inversion du modèle avec une marge de saturation soit le système optimal pour cette tâche, et une autre fonction pourrait permettre d'avoir de meilleures performances.

Dans ce paragraphe nous avons montré qu'un correcteur basé sur le modèle de l'amplificateur est efficace, et donc qu'il est possible (à condition de connaître parfaitement la réponse de l'amplificateur) de compenser les non-linéarités dans un système OFDM avec un correcteur dans le domaine temporel. Cependant nous venons de voir la mise en pratique de ce correcteur théorique présente des difficultés, et nous allons maintenant étudier sa réalisation à l'aide de réseaux de neurones.

3. Égalisation à l'aide d'un PMC

Les deux courbes de la figure 4.9 nous serviront de référence, et les simulations seront effectuées avec un système OFDM à 48 porteuses et une modulation MAQ16. Une base d'apprentissage est constituée, mais cette fois-ci avec des échantillons temporels du symbole OFDM. Pour les expériences suivantes, le rapport signal sur bruit du canal est de $E_b/N_0 = 13\mathrm{dB}$ dans la base d'apprentissage.

Les PMC possèdent deux entrées et deux sorties, car les signaux qu'ils traitent sont complexes. Différentes architectures ont été testées, avec tout d'abord une couche cachée de 2 neurones, puis en augmentant progressivement le nombre de neurones, avant de tester un réseau avec deux couches cachées, également en augmentant progressivement le nombre de neurones sur les deux couches. Les fonctions d'activation des couches cachées sont des tangentes hyperboliques, et celles des couches de sortie sont linéaires. Nous avons constitué une base de moins 2048 symboles, et évité un surapprentissage avec toutes les structures que nous avons testé. La figure 4.10 montre le taux d'erreur pour quelques correcteurs. Comme dans le chapitre précédent, PMC-x représente un PMC avec une couche cachée de x neurones, PMC-x-y un PMC avec deux couches cachées de x et y neurones.

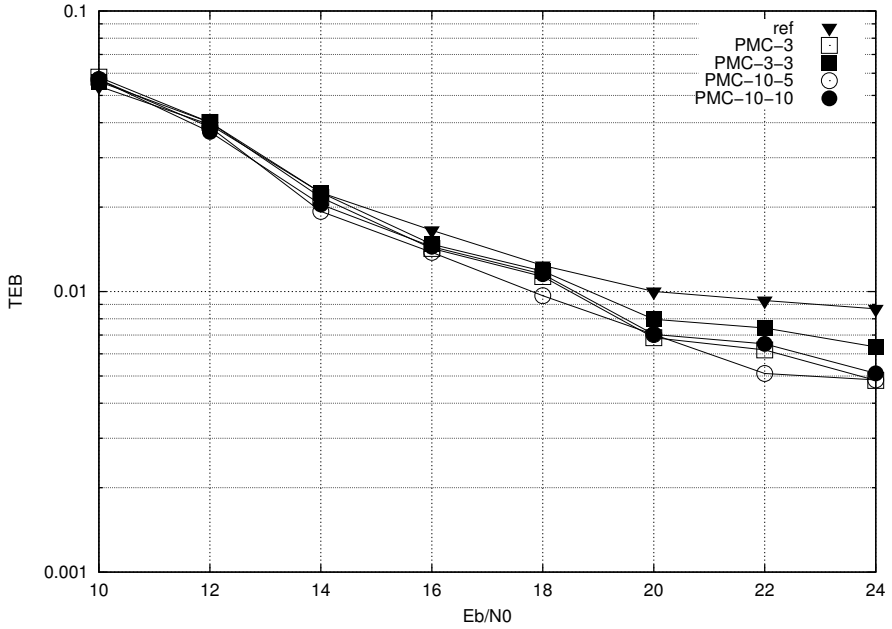

Figure 4.10. Taux d'erreur binaire d'une chaîne OFDM avec correcteur PMC temporel, 48 porteuses, une modulation MAQ16, et un amplificateur SSPA avec un recul de 0 dB

Tous les correcteurs, y compris le plus simple présenté, réalisé avec un PMC constitué d'une couche cachée de 3 neurones, parviennent à réduire le taux d'erreur binaire par rapport au système de référence. Celui qui obtient les meilleures performances est le PMC-10-5. Si l'on regarde sa courbe d'apprentissage (figure 4.11), on constate que contrairement au cas fréquentiel, l'algorithme d'apprentissage (ici un Levenberg Marquardt) converge, et dès l'itération 25 l'erreur quadratique moyenne ne diminue presque plus.

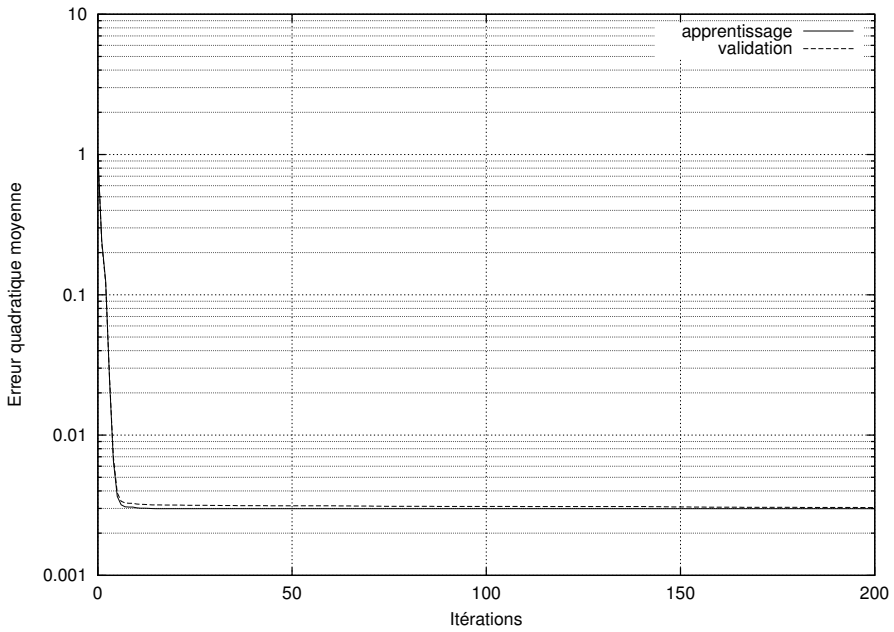

Figure 4.11. Courbe d'apprentissage du PMC-10-5

Cependant même ce correcteur ne parvient pas à obtenir les mêmes performances que le modèle SSPA inversé (courbe 'sspa'), comme le montre la figure 4.12 :

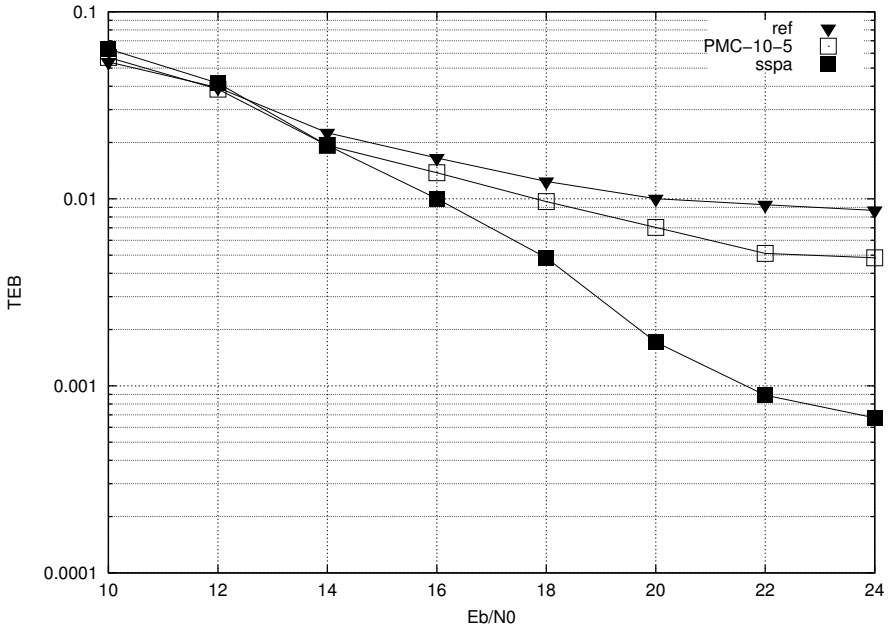

Figure 4.12. Comparaison entre le taux d'erreur binaire avec le correcteur PMC-10-5 et celui avec le correcteur SSPA inversé dans un système OFDM à 48 porteuses, un amplificateur SSPA avec un recul de 0 dB et une modulation MAQ16

Nous avons testé d'autres PMC avec plus de neurones, réalisé des apprentissages avec différents niveaux de bruits, mais nous ne sommes pas parvenu à approcher la courbe obtenue par le correcteur SSPA. D'autres architectures neuronales, comme les GRBF ou les réseaux d'ordre supérieur, pourraient peut-être mieux convenir à cette application que les PMC. Nous nous sommes en particulier intéressés une fois de plus aux réseaux d'ordre supérieur.

4. Égalisation à l'aide d'un RPN

4.1. Mise en oeuvre

Les correcteurs à RPN sont utilisés de la même manière que les correcteurs précédents à PMC, la seule différence étant le réseau de neurones utilisé. Les RPN ont donc également deux entrées et deux sorties. La fonction d'activation du réseau est également une tangente hyperbolique. L'ensemble des valeurs que peuvent atteindre les sorties du réseau RPN est défini par la fonction d'activation, c'est à dire $[-1, 1]$ dans le cas de la tangente hyperbolique. Cet intervalle peut être insuffisant suivant la puissance des signaux traités et les différents gains du système. Dans notre application nous avons ajouté un gain de 3 en sortie du réseau pour étendre l'amplitude maximale possible du signal. Avec une fonction d'activation linéaire il n'est pas nécessaire de faire un tel ajustement, et les résultats obtenus dans ce dernier cas seront montrés en section 4.3 p. 112.

Afin de déterminer le meilleur rapport signal sur bruit pour l'apprentissage du RPN, nous avons procédé comme lors de l'expérience présentée figure 3.24, c'est à dire que nous traçons des courbes de taux d'erreur binaire à rapport signal sur bruit dans le canal constant, pour différents rapports signal sur bruit d'apprentissage (en abscisse). Tous les RPN utilisés sont d'ordre 5, avec comme fonction d'activation une tangente hyperbolique multipliée par un facteur 3.

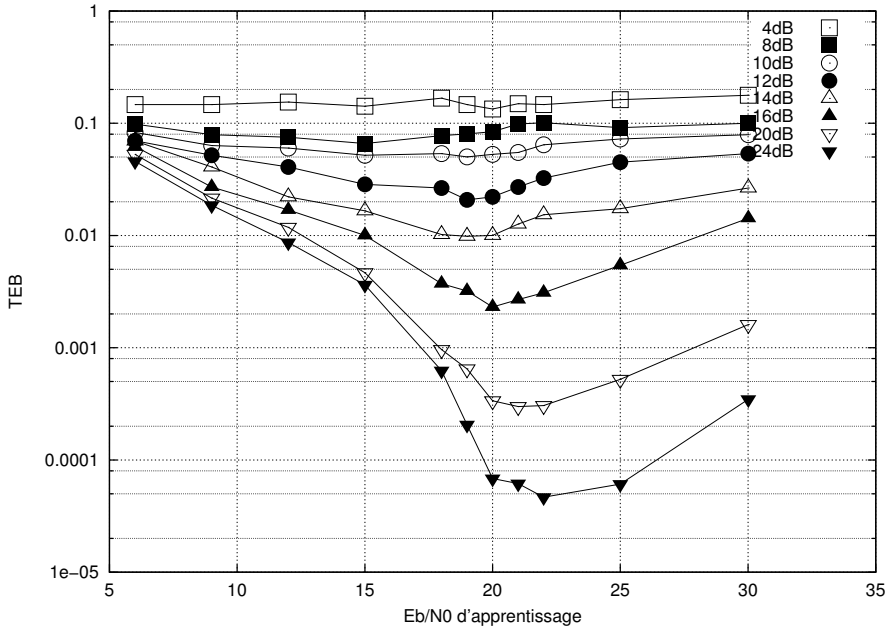

Figure 4.13. Taux d'erreur binaire d'un système OFDM à 48 porteuses (amplificateur SSPA, recul de 0 dB, modulation MAQ16) avec correcteur RPN temporel. Courbes à Eb/N0 fixe dans le canal, apprentissage avec différents Eb/N0

Chaque courbe représente une simulation avec un rapport E_b/N_0 fixe dans le canal, avec différents réseaux RPN dont on indique en abscisse le rapport E_b/N_0 qui a servi à constituer la base d'apprentissage. Sur chaque courbe l'on cherche le minimum, qui indique alors la base d'apprentissage à sélectionner pour avoir les meilleures performances. Lorsque le rapport E_b/N_0 est faible dans le canal, on constate que ce minimum se situe à un E_b/N_0 d'apprentissage légèrement inférieur à 20 dB. Ensuite pour des rapports E_b/N_0 plus élevés dans le canal, ce minimum se situe à un E_b/N_0 d'apprentissage légèrement supérieur à 20 dB. Pour faire un

compromis entre ces deux tendances nous avons choisi un E_b/N_0 d'apprentissage de 20 dB, qui a de bonnes performances dans la plupart des cas.

4.2. Performances du correcteur

La figure 4.14 montre la courbe d'apprentissage du réseau RPN dans ces conditions retenues (rapport signal sur bruit de 20 dB dans la base d'apprentissage) en augmentant son ordre à 7 :

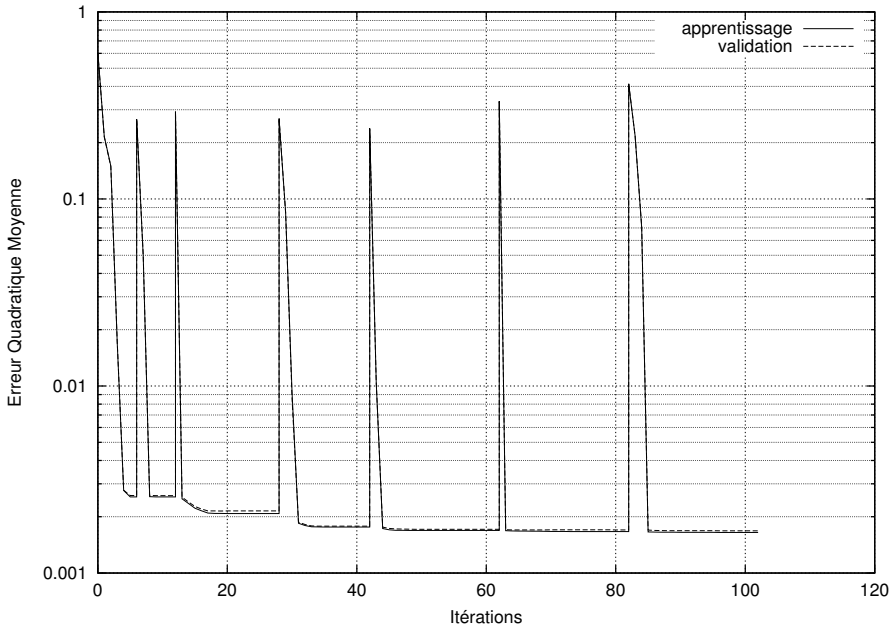

Figure 4.14. Apprentissage d'un réseau RPN pour une correction temporelle. Eb/N0 d'apprentissage de 20 dB, système OFDM à 48 porteuses avec un amplificateur SSPA, un recul de 0 dB et une modulation MAQ16

On constate que les courbes de performance sur les bases de validation et d'apprentissage sont quasiment confondues, ce qui confirme l'absence de surapprentissage. De plus on constate que l'erreur quadratique moyenne ne diminue plus lors de l'apprentissage des ordres 6 et 7, et c'est pourquoi nous nous sommes limités à l'ordre 5. L'apprentissage jusqu'à l'ordre 5 prend 30 secondes sur une station Sun Ultra 10. La figure 4.15 montre la courbe d'erreur binaire obtenue avec ce réseau limité à l'ordre 5 ('rpn'), avec en comparaison le système de référence sans correcteur ('ref'), celui avec le correcteur à PMC-10-5 ('pmc') et le correcteur SSPA inversé avec une marge de saturation de 0,15 dB ('sspa') :

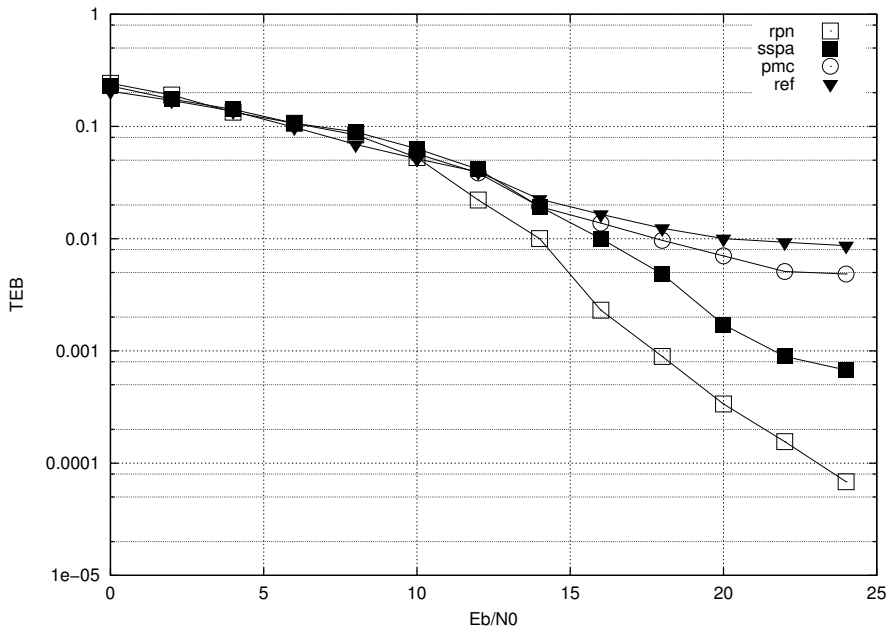

Figure 4.15. Comparaison entre le taux d'erreur binaire avec le correcteur RPN temporel et celui avec le correcteur SSPA inversé, dans un système OFDM à 48 porteuses avec un amplificateur SSPA, un recul de 0 dB et une modulation MAQ16

Le correcteur RPN a cette fois de meilleures performances que le correcteur théorique SSPA, et que le PMC. Pour un taux d'erreur binaire de 10^{-2}, le gain en rapport signal sur bruit est de 4 dB avec le correcteur SSPA et de 6 dB avec le correcteur RPN. Pour un rapport E_b/N_0 de 20 dB le taux d'erreur binaire est divisé par 6 avec le correcteur SSPA et par 30 avec le correcteur RPN. La différence entre les correcteurs peut s'expliquer en comparant leur réponse en module. Dans la figure 4.16 ces réponses sont tracées avec le module de l'entrée en abscisse et celui de la sortie en ordonnée (la courbe 'sspa' montre la réponse du correcteur sans marge de saturation) :

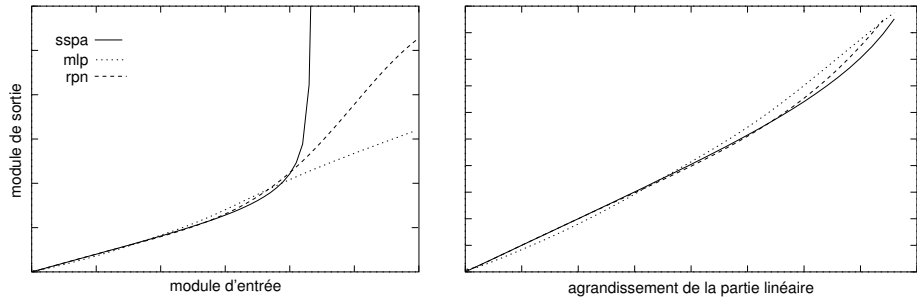

Figure 4.16. Réponses des correcteurs SSPA, RPN et PMC-10-5

La partie de droite est un agrandissement de la partie quasi-linéaire des courbes. Sur la figure de gauche, on constate que les trois correcteurs ont des réponses très proches dans la partie linéaire, puis lorsque la courbe du correcteur SSPA prend une pente très importante, celles des réseaux de neurones ont une pente plus modérée. Si l'on regarde plus en détail la partie linéaire, sur la figure de droite, on constate que le PMC s'écarte plus de la courbe idéale que le RPN. Ceci peut expliquer la moins bonne performance du correcteur PMC, qui doit introduire des erreurs supplémentaires lorsque le signal OFDM est en dessous du seuil de saturation de l'amplificateur. Le réseau RPN parvient à établir un meilleur compromis entre les parties linéaire et non linéaire de la courbe à approximer.

4.3. Rôle de la fonction d'activation

Afin de qualifier l'influence de la fonction d'activation sur les performances du correcteur RPN, nous avons comparé ses performances avec deux fonctions d'activations différentes : la tangente hyperbolique, avec éventuellement un gain (le correcteur qui a été présenté jusque là) et une fonction linéaire. Dans le second cas, la fonction réalisée par le réseau RPN sur chacune de ses sorties est un polynôme à deux variables.

Les deux réseaux ont été entraînés avec la même base d'apprentissage, avec un rapport signal sur bruit de 20 dB. La figure 4.17 montre les courbes de taux d'erreur binaire obtenus avec les deux correcteurs :

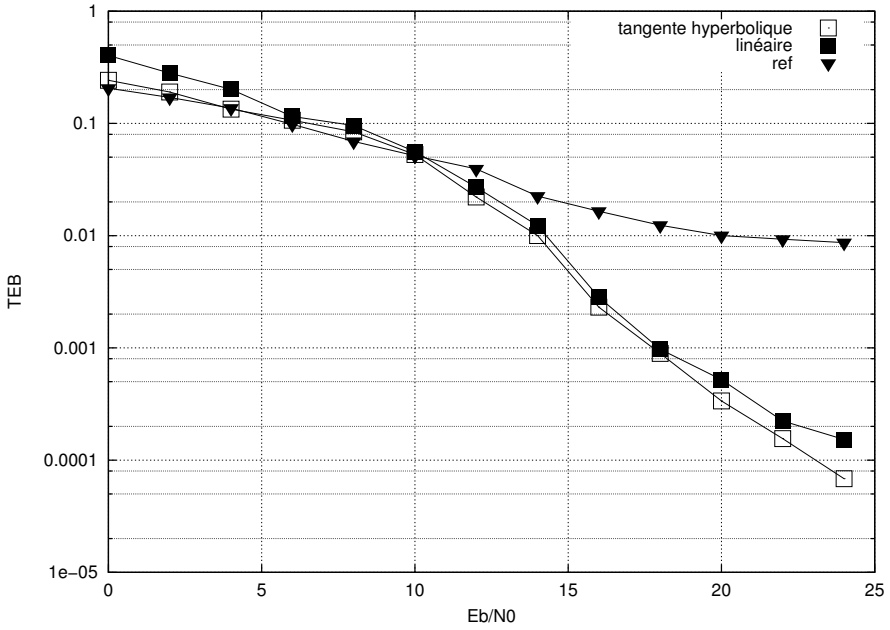

Figure 4.17. Comparaison entre le taux d'erreur binaire avec le correcteur RPN à tangente hyperbolique et celui avec une fonction linéaire, dans un système OFDM à 48 porteuses avec un amplificateur SSPA, un recul de 0 dB et une modulation MAQ16

Les résultats obtenus avec les deux correcteurs sont très proches, mais celui avec la tangente hyperbolique est légèrement meilleur. Une des raisons possibles est que la tangente hyperbolique permet au réseau RPN d'approximer plus facilement la fonction voulue avec moins de paramètres.

4.4. Autres non linéarités

Afin de vérifier que le réseau de neurones est capable de s'adapter à d'autres types de non linéarité, nous l'avons également testé avec le modèle SSPA en prenant le paramètre $p = 3$ dans l'équation (3.2). Le réseau a été entraîné avec une base d'apprentissage ayant un rapport signal sur bruit de $E_b/N_0 = 20\mathrm{dB}$, et la figure 4.18 montre la courbe de taux d'erreur binaire obtenue :

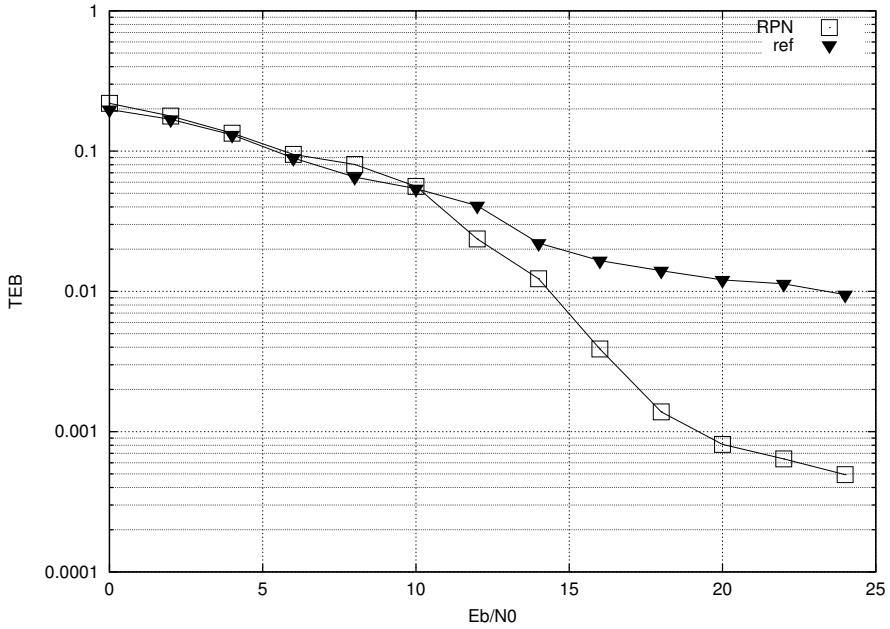

Figure 4.18. Taux d'erreur binaire d'un système OFDM à 48 porteuses avec modèle d'amplificateur SSPA3, un recul de 0 dB, une modulation MAQ16 et un correcteur RPN temporel

Dans ce cas également, le réseau apporte une diminution importante du taux d'erreur binaire. Le gain pour un taux d'erreur binaire de 10^{-2} est d'environ 8 dB et pour un rapport E_b/N_0 de 20 dB le taux d'erreur binaire est divisé par 15. Enfin on peut mesurer les performances du correcteur dans le cas d'un limiteur, même si dans ce cas on ne s'attend pas à mesurer de gain apporté par le correcteur. En effet la réponse de l'amplificateur n'est pas inversible. La figure 4.19 montre les résultats obtenus :

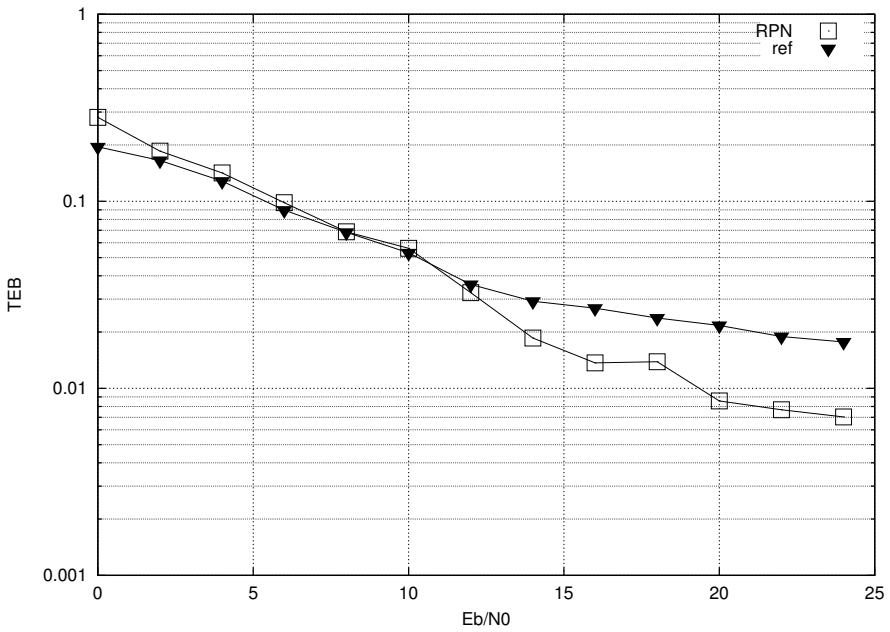

Figure 4.19. Taux d'erreur binaire d'un système OFDM à 48 porteuses avec limiteur, un recul de 0 dB, une modulation MAQ16 et un correcteur RPN temporel

Le gain apporté par le réseau de neurones n'est pas important, mais il est réel. Le taux d'erreur binaire est divisé par 3 pour un rapport E_b/N_0 de 20 dB. La figure 4.20 montre la réponse en module de ce correcteur :

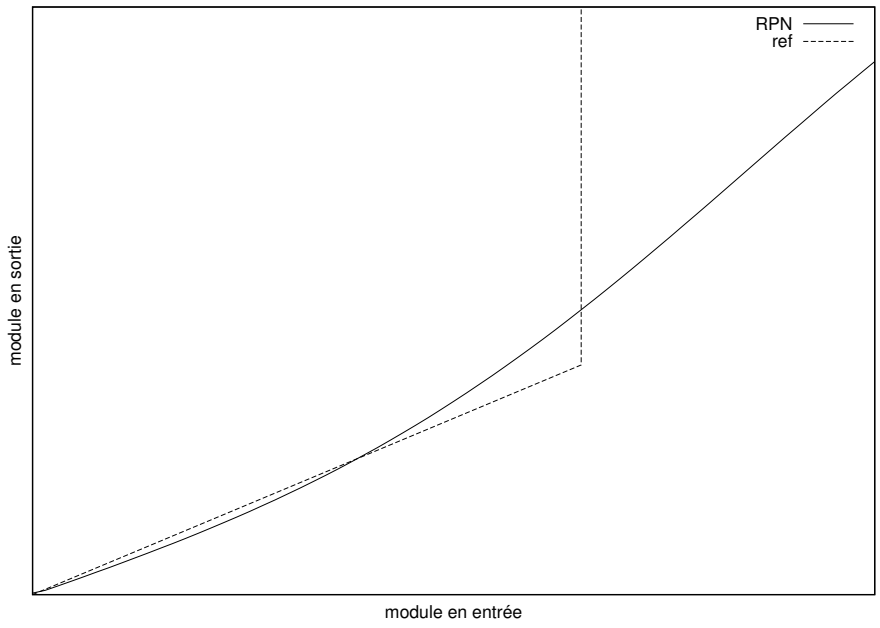

Figure 4.20. Réponse en module du correcteur RPN dans le cas d'un limiteur

En amplifiant légèrement le signal avant le seuil de saturation, le correcteur à RPN doit permettre de réduire le taux d'erreur. Cependant si l'on compare les résultats présentés figure 4.19 avec les résultats précédents, il apparaît que correcteur RPN temporel n'est pas adapté au limiteur, car la diminution du taux d'erreur est relativement faible.

5. Simplification du correcteur RPN

Nous avons vu que le RPN permet d'obtenir de bonnes performances dans le cas d'un système avec amplificateur SSPA. Cependant ce modèle d'amplificateur n'agit que sur le module du signal à corriger, et laisse inchangé la phase. Nous avons donc fait un correcteur simplifié qui lui aussi n'agit que sur le module du signal :

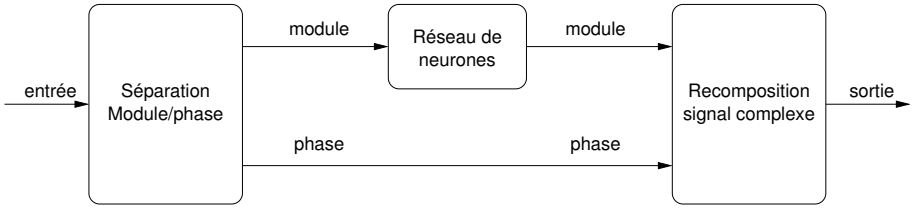

Figure 4.21. Principe du correcteur simplifié

Le réseau de neurones n'a ainsi plus qu'une entrée et une sortie. Dans ces conditions le RPN demande beaucoup plus de coefficients qu'un HPU pour réaliser un polynôme de même degré, et donc un HPU (décrit en section 4.3 p. 60), toujours avec une tangente hyperbolique comme fonction d'activation, a été simulé. Pour un HPU de degré 5, la fonction réalisée est alors de la forme :

$$f(x) = tanh(a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5) \tag{4.2}$$

Ce système ne comporte plus que 6 paramètres. Ce correcteur simplifié a été testé dans une chaîne OFDM avec 48 porteuses, une modulation MAQ16 et un amplificateur SSPA. La figure 4.22 montre les courbes d'erreurs obtenues avec différentes tailles de la base ayant servi à l'algorithme d'apprentissage, celui de Levenberg Marquardt.

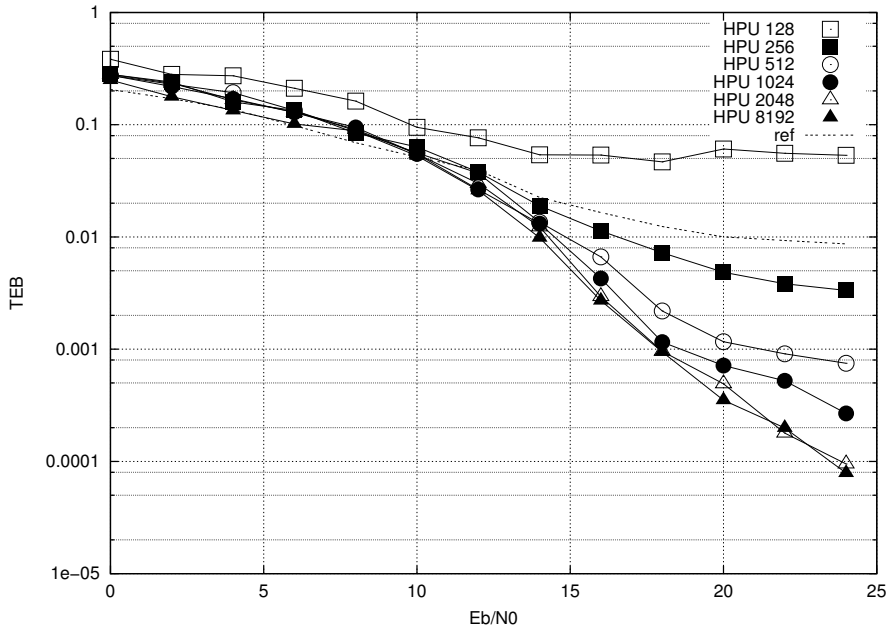

Figure 4.22. Taux d'erreur binaire avec le correcteur temporel simplifié, avec différentes tailles de la base d'apprentissage, dans un système OFDM à 48 porteuses et un amplificateur SSPA, un recul de 0 dB et une modulation MAQ16

A partir de 2048 éléments le correcteur HPU a une performance équivalente à celle du correcteur RPN. On note ainsi un gain de 6 dB pour un taux d'erreur binaire de 10^{-2}. Pour des bases de plus de 2048 éléments, le taux d'erreur binaire ne diminue plus de manière significative. L'apprentissage de ce HPU par un algorithme de Levenberg-Marquardt prend 8 secondes sur une station Sun Ultra 10, ce qui est presque 4 fois plus rapide que celui du RPN.

Le correcteur ainsi formé a des performances équivalentes à celles du RPN, tout en étant plus simple et nécessitant moins de calcul pour l'apprentissage. Ses résultats ont été publiés dans [TERT03b]. Cependant ce correcteur est moins généraliste que le RPN, puisqu'il n'agit que sur le module du signal. Un autre modèle d'amplificateur non linéaire présenté dans [RAPP91], le tube à ondes progressives, présente une non linéarité de phase en plus de la non linéarité d'amplitude. Le correcteur à RPN devrait pouvoir compenser ce type d'amplificateur sans modifications, alors qu'avec le correcteur simplifié, il faudrait ajouter un second HPU qui serait chargé de compenser la non-linéarité de phase.

Conclusion

Dans ce chapitre nous avons présenté une seconde méthode pour compenser les non-linéarités de l'amplificateur dans un récepteur OFDM. La compensation est cette fois réalisée dans le domaine temporel, ce qui permet d'utiliser des réseaux de neurones plus simples. Dans un premier temps un correcteur simple basé sur le modèle inverse de l'amplificateur non linéaire a été simulé, et il a été établi qu'il est efficace à condition d'y ajouter une marge de saturation. Cependant il nécessite de connaître parfaitement le modèle de l'amplificateur, ce qui est difficile à garantir en pratique.

La méthode choisie, à l'aide de réseaux de neurones, est inspirée de techniques d'égalisation non linéaire dans le contexte de modulations monoporteuses. Nous avons montré d'une part que cette méthode peut s'appliquer au cas de l'OFDM et d'autre part qu'un réseau d'ordre supérieur permet de meilleures performances que le PMC. Il serait intéressant de tester ces correcteurs dans un modulateur monoporteuse afin de voir si un réseau d'ordre supérieur pourrait également avoir de meilleures performances dans ces systèmes.

Comme pour le correcteur fréquentiel, il existe un rapport signal sur bruit optimal pour la constitution de la base d'apprentissage afin d'obtenir les meilleures performances en exploitation. Avec un RPN nous avons obtenu un gain de 6 à 8 dB pour un taux d'erreur binaire de 10^{-2}, suivant le modèle d'amplificateur. En pratique cela veut dire qu'il est possible d'utiliser un amplificateur 4 fois moins puissant dans l'émetteur tout en conservant un taux d'erreur binaire de 10^{-2}, en ajoutant ce correcteur dans le récepteur. Enfin nous avons présenté un correcteur simplifié, adapté aux amplificateurs de type SSPA, qui est basé sur un réseau HPU et qui obtient les mêmes performances que le correcteur RPN, tout en étant plus simple aussi bien au niveau de l'apprentissage que de l'exploitation.

A présent nous pouvons aborder la mise en oeuvre des correcteurs présentés, et établir quelques éléments de comparaison des différentes approches entre elles et avec d'autres techniques de compensation des non linéarités.

Chapitre 5. Mise en oeuvre et comparaison des différentes approches

Introduction

Dans les deux chapitres précédents, nous avons présenté deux méthodes de compensation des non-linéarités à l'aide de réseaux de neurones dans le cadre de l'OFDM, dans les domaines fréquentiel et temporel. Ce dernier chapitre aborde d'une part certains aspects pratiques de ces correcteurs, tels que leur apprentissage et leur comportement en présence d'un canal multitrajets, et d'autre part des comparaisons entre les deux méthodes, ainsi qu'une comparaison avec une autre solution au problème des non-linéarités en OFDM proposée dans la littérature.

1. Mise en oeuvre

1.1. Performances sur un canal multitrajet

Dans les chapitres précédents les correcteurs ont été simulés sur des canaux présentant un seul trajet sans atténuation ni retard, et un bruit blanc additif gaussien. Or un des intérêts de la modulation OFDM est sa grande adaptation aux canaux multitrajets. Dans les deux systèmes présentés l'égalisation du canal est indépendante de la compensation de la non-linéarité faite par le réseau de neurones. Le correcteur devrait donc avoir les mêmes performances quel que soit le canal dans lequel se propage le signal OFDM. Nous allons vérifier cette affirmation à l'aide d'une simulation sur un canal multitrajet.

Le modèle de canal que nous avons choisi pour cette simulation est un modèle présent dans le logiciel SPW, et proposé dans [SALE87] comme modèle de canal Indoor, c'est à dire à l'intérieur d'un bâtiment. Ce modèle prend en compte deux phénomènes. Tout d'abord les réflexions sur les objets à proximité immédiate de l'émetteur et du récepteur vont causer plusieurs échos très proches les uns des autres. Cet ensemble d'échos très proches est appelé paquet. Ensuite les obstacles entre les deux appareils radio (les murs et les portes par exemple) vont causer d'autres échos, mais plus espacés entre eux. Saleh propose donc dans son modèle de créer tout d'abord un paquet composé d'échos très proches, avec une distribution uniforme des déphasages et une distribution de Rayleigh pour les amplitudes. Ensuite la réponse du canal est composée de plusieurs paquets, chaque paquet étant une copie retardée et atténuée du premier. Le retard de chaque paquet est déterminé à l'aide une loi de Poisson.

Pour paramétrer le modèle nous avons utilisé les valeurs suggérées dans le modèle de Saleh dans de SPW [CADE02] qui simule un canal à l'intérieur d'un immeuble dans la bande des 1,5 GHz. Le canal a une réponse impulsionnelle avec un étalement des retards τ_R de 600 ns, avec un temps moyen entre paquets de 200 ns et des échos dans chaque paquet espacés en moyenne de 5 ns. La figure montre le module de la réponse impulsionnelle de ce canal :

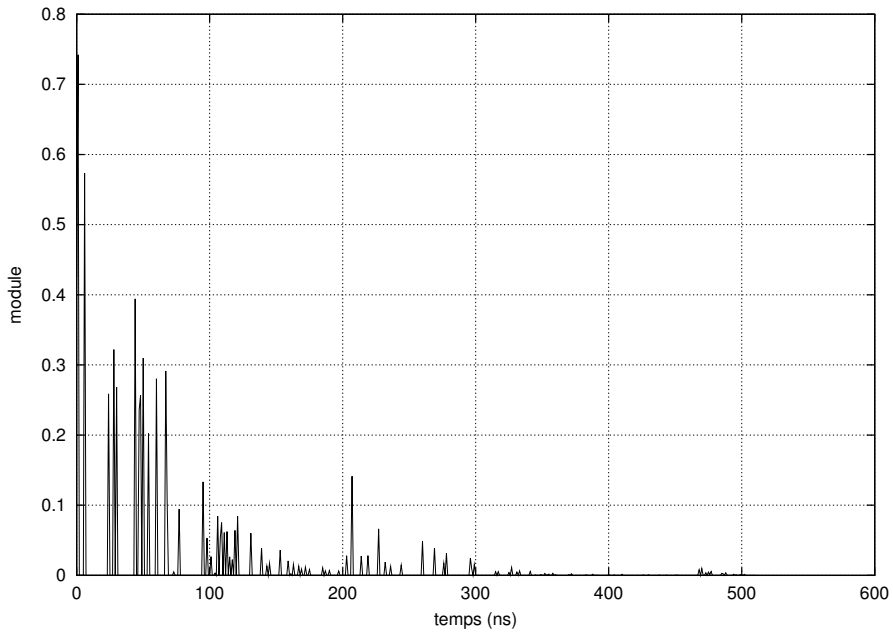

Figure 5.1. Module normalisé en puissance de la réponse impulsionnelle du canal multitrajets

Pour la modulation OFDM nous avons choisi une fréquence d'échantillonnage de 125 MHz, ce qui veut dire que la réponse du canal a une durée de 75 échantillons temporels. Nous avons donc fixé la taille de l'intervalle de garde à 128 échantillons et la longueur du symbole OFDM temporel à 512 échantillons. En effet, comme rappelé dans la section 2.4 p. 26, la durée de l'intervalle de garde doit être supérieur à l'étalement des retards, et il est conseillé qu'elle soit environ le quart de la durée du symbole OFDM. Le nombre de porteuses est donc de 512, et nous avons choisi une modulation MAQ16. Le correcteur utilisé est un RPN dans le domaine temporel, étant donné que nous n'avons pas réussi à entraîner correctement un RPN dans le domaine fréquentiel avec une modulation OFDM ayant plus de 8 porteuses.

L'égalisation du canal et l'apprentissage du RPN utilisent comme base d'apprentissage des symboles OFDM connus de l'émetteur et du récepteur. L'égaliseur de canal que nous avons utilisé est très simple, il estime la réponse fréquentielle du canal en divisant les symboles reçus sur chaque porteuse par le symbole connu. Le cas non linéaire peut poser un problème car les symboles reçus sont déformés par la non-linéarité, ce qui n'est pas le cas des symboles connus. Cependant on constate en pratique que malgré cette différence, cet égaliseur simple parvient à estimer correctement le canal. On peut quantifier la précision de l'estimation du canal par le critère suivant :

$$\frac{\|\mathbf{c_e} - \mathbf{c_c}\|^2}{\|\mathbf{c_c}\|^2} \tag{5.1}$$

où $\mathbf{c_c}$ est la réponse du canal et $\mathbf{c_e}$ celle estimée par le correcteur. Ces vecteurs sont des vecteurs complexes, de dimension N_p (le nombre de porteuses) et dont chaque composante est la réponse fréquentielle (estimée ou réelle) du canal à la fréquence d'une porteuse. Si l'on trace l'évolution de ce critère en fonction du nombre de symboles OFDM qui servent à évaluer la réponse du canal, figure 5.2, on constate que l'égaliseur parvient à une estimation correcte au bout d'environ 50 symboles. Nous avons pris une marge en fixant le nombre de symboles OFDM de calibration à 80.

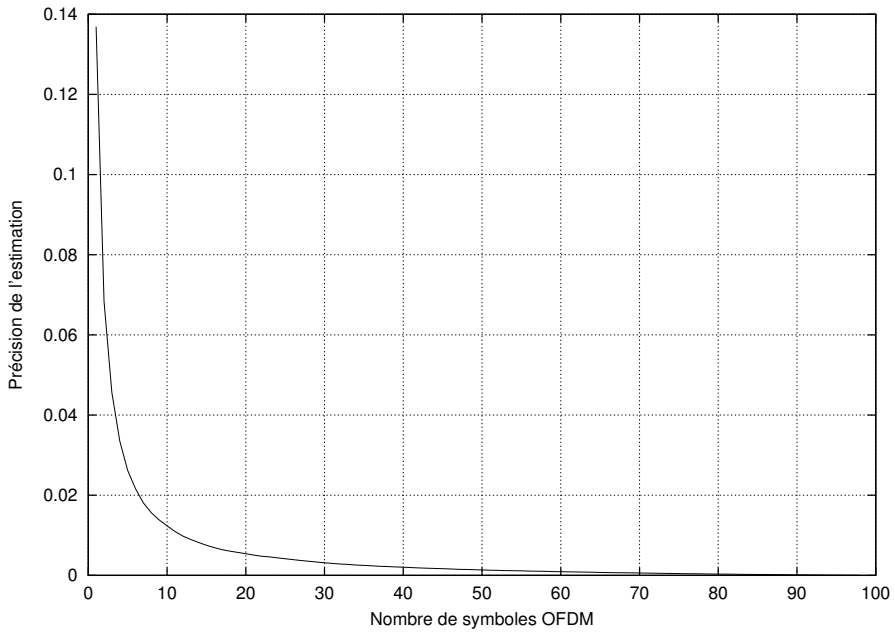

Figure 5.2. Évolution de l'estimation du canal

Une fois le canal égalisé, l'apprentissage du réseau RPN peut avoir lieu. La figure 5.3 montre le taux d'erreur binaire du système obtenu. 'ref' est la courbe sans correcteur et 'rpn' est celle avec le réseau RPN.

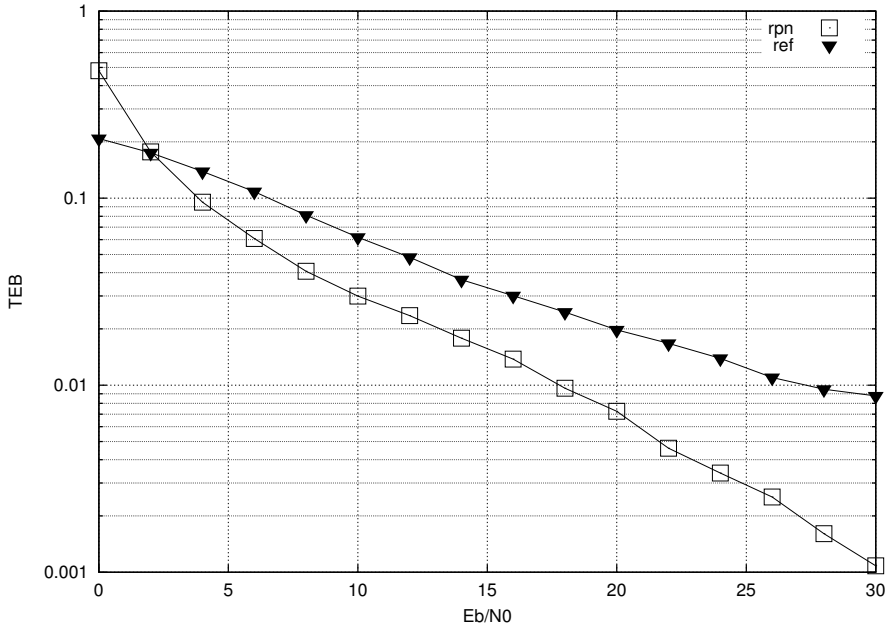

Figure 5.3. Taux d'erreur binaire en fonction du rapport signal sur bruit. Système OFDM à 512 porteuses, amplificateur SSPA avec un recul de 0 dB, une modulation MAQ16 et un canal multitrajet

Le gain apporté par le correcteur pour un taux d'erreur binaire de 10^{-2} est ici de presque 10 dB et pour un rapport E_b/N_0 de 20 dB ce taux d'erreur binaire est divisé par 3. A titre de comparaison voici la même courbe sur un canal de Gauss (monotrajet) :

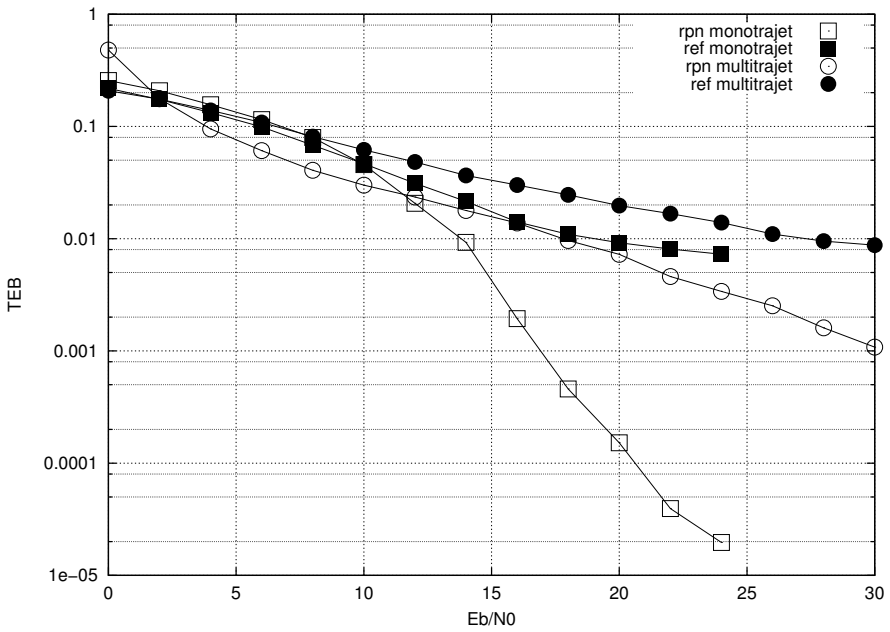

Figure 5.4. Taux d'erreur binaire en fonction du rapport signal sur bruit. Système OFDM à 512 porteuses, amplificateur SSPA avec un recul de 0 dB, une modulation MAQ16 et un canal de Gauss, et comparaison avec les courbes obtenues figure précédente

Le fait que le canal soit multitrajets augmente le taux d'erreur binaire, mais l'on constate que dans les deux cas le réseau de neurones apporte un gain significatif au système. Le correcteur préserve donc l'avantage de l'OFDM sur les canaux multitrajets.

1.2. Implémentation et apprentissage

Les deux correcteurs présentés nécessitent une certaine puissance de calcul. Il faut donc placer dans le récepteur un processeur capable d'effectuer toutes les opérations demandées. Mais comme une réception OFDM demande déjà un grand nombre de calculs (notamment pour la transformée de Fourier) un tel calculateur est déjà disponible. L'implémentation de ces calculateurs nécessitera donc simplement une puissance de calcul supérieure. Une étude plus quantitative en terme de puissance de calcul nécessaire aux deux correcteurs sera effectuée dans la section 2.2 p. 122.

L'apprentissage du correcteur peut être réalisé de deux manières différentes. Une première manière est de faire cet apprentissage par le récepteur lors de l'établissement de la communication. L'émetteur transmet un certain nombre[9] de symboles OFDM connus de lui et du récepteur, et ceci constitue une base d'apprentissage. Cette approche a l'avantage d'une plus grande souplesse d'utilisation, le récepteur pouvant s'adapter à n'importe quel amplificateur, à n'importe quel moment. Mais elle nécessite plus de calcul, surtout pour le correcteur dans le domaine fréquentiel. L'apprentissage du correcteur RPN temporel sur un système à 48 porteuses demande 30 secondes sur une station de travail Sun Ultra 10, celui du correcteur HPU dans les mêmes conditions 8 secondes, mais il risque de demander plus de temps dans le cas

[9] ce nombre dépend de l'application. Dans la simulation sur un canal multitrajet présentée section 1.1 p. 117, l'égalisation du canal a demandé 80 symboles OFDM, et l'apprentissage du réseau de neurones 2048 échantillons temporels, soit 4 symboles OFDM. Le temps de transmission de l'ensemble de ces symboles à la fréquence d'échantillonnage de 125MHz est de 0,43ms, intervalles de garde compris

d'un système embarqué qui possède généralement une puissance de calcul moindre. De plus nous avons vu qu'il y avait un rapport signal sur bruit optimal dans chaque cas. Si le rapport signal sur bruit du canal ne correspond pas à ce rapport optimal, le réseau obtenu n'aura pas les meilleures performances.

La seconde solution est d'effectuer l'apprentissage une fois, et de mémoriser les paramètres du correcteur obtenu dans l'émetteur. Ensuite ces paramètres sont envoyés au récepteur, qui initialisera son correcteur avec les paramètres reçus. Ces paramètres devront être envoyés avec un code correcteur d'erreurs plus puissant que celui employé pour le reste de la communication car le récepteur ne peut pas encore compenser les non-linéarités au moment où il les reçoit. Cette méthode a l'avantage de simplifier le récepteur, qui n'a plus besoin de réaliser l'apprentissage, mais a le défaut de modifier le protocole de communication. De plus dans le cas d'une diffusion (un signal vidéo par exemple) l'émetteur devra envoyer régulièrement ces paramètres, et donc une partie du débit utile sera perdue pour ceux-ci.

2. Comparaison des deux approches proposées

2.1. Performances

Le correcteur dans le domaine fréquentiel n'a pu être efficacement entraîné que dans le cas d'un système à 4 porteuses. Avec 8 porteuses l'apprentissage commence à être difficile, et s'avère inefficace à 16 porteuses. Pour comparer les deux correcteurs nous avons donc pris un système à 4 porteuses, et analysé le taux d'erreur binaire dans les deux cas. Dans la figure 5.5, 'ref' indique le taux d'erreur binaire du système sans correcteur, 'rpn-f' celui obtenu avec un correcteur RPN dans le domaine fréquentiel, et 'rpn-t' celui dans le domaine temporel.

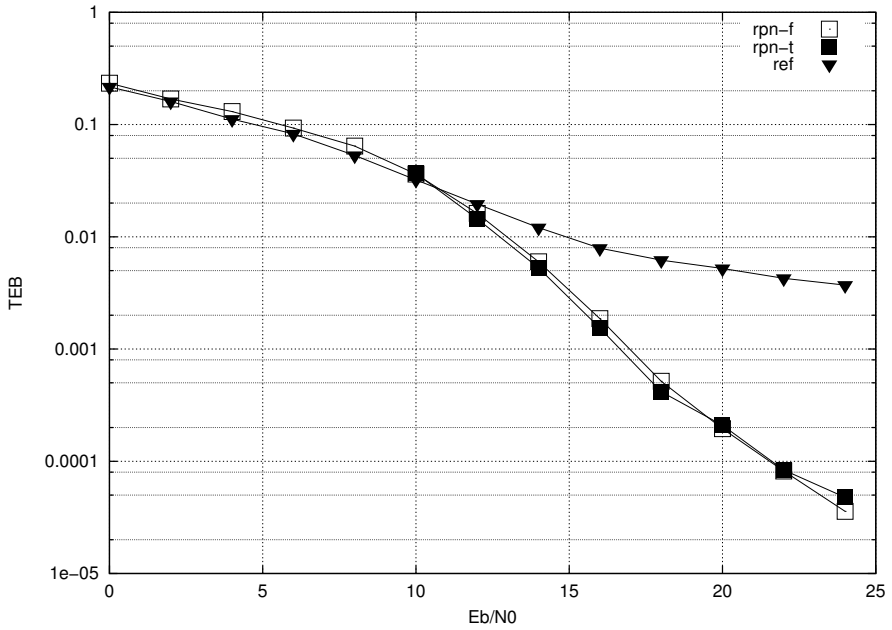

Figure 5.5. Taux d'erreur binaire en fonction du rapport signal sur bruit dans un système OFDM avec 4 porteuses, un amplificateur SSPA, un recul de 0 dB, une modulation MAQ16 et les correcteurs RPN fréquentiel et temporel

Les taux obtenus dans les deux cas sont quasiment identiques, les deux correcteurs ont les mêmes performances. Cependant il faut noter que sans amélioration de l'algorithme

d'apprentissage, le correcteur temporel est actuellement le seul qui a de bonnes performances avec un nombre plus élevé de porteuses.

2.2. Puissance de calcul nécessaire

Le correcteur temporel utilise un réseau de neurones moins complexe que le correcteur fréquentiel, car il possède moins d'entrées. Mais il demande plus de calculs avant et après le réseau, puisqu'il utilise deux transformées de Fourier supplémentaires. On peut donc supposer que pour un petit nombre de porteuses le correcteur fréquentiel nécessitera moins de calcul, mais pour un nombre plus grand de porteuses c'est le correcteur temporel qui sera le plus simple. Afin de vérifier cette hypothèse, nous allons calculer le nombre d'opérations nécessaires dans chaque cas pour corriger un symbole OFDM.

Dans ce calcul, nous allons supposer que toutes les opérations élémentaires (multiplication, division, addition, soustraction) ont la même durée. Ceci n'est pas forcément le cas en pratique, mais permet d'avoir une idée de la complexité des différents algorithmes. Nous allons commencer par calculer le nombre d'opérations élémentaires dans chaque étape de calcul. On rappelle que N_p est le nombre de porteuses.

2.2.1. Calculs de complexité des différents éléments

La transformée de Fourier rapide est un algorithme qui permet de calculer avec moins d'opérations une transformée de Fourier discrète dans le cas où le nombre d'éléments est une puissance de 2 [BELL02]. Dans ce cas, la transformée est calculée en $\log_2 N_p$ étapes, composées chacune de $N_p/2$ multiplications complexes et N_p additions complexes, soit $2N_p$ multiplications et $3N_p$ additions réelles. Le nombre d'opérations élémentaires est donc :

$$5N_p \log_2 N_p \tag{5.2}$$

Si N_p n'est pas une puissance de 2, nous l'arrondirons à la puissance de 2 supérieure.

L'égalisation linéaire se fait dans le domaine fréquentiel, avec N_p multiplications complexes, soit $4N_p$ multiplications réelles et $2N_p$ additions réelles. Le nombre d'opérations élémentaires est donc :

$$6N_p \tag{5.3}$$

Soit un réseau RPN avec N_e entrées réelles, N_s sorties réelles et d'ordre O_r. Ce réseau est composés de N_s réseaux RPN qui calculent chacun une sortie. Chaque réseau RPN est composé de O_r réseaux Pi-Sigma, d'ordres $i = 1 \ldots O_r$. Enfin un réseau Pi-Sigma d'ordre i est constitué de i neurones Sigma, qui effectuent chacun un produit scalaire entre l'entrée et un vecteur poids, et y ajoutent un biais. Donc un neurone sigma nécessite N_e multiplications et N_e additions. Le nombre d'opérations d'un neurone Sigma est donc :

$$2N_e \tag{5.4}$$

Un réseau Pi-Sigma d'ordre i effectue le produit des sorties de ses neurones Sigma, soit $i - 1$ multiplications supplémentaires. Le nombre d'opérations d'un réseau Pi-Sigma est donc :

$$2iN_e + i - 1 \tag{5.5}$$

Un réseau RPN effectue la somme des sorties de ses réseaux Pi-Sigma. Son nombre d'opérations élémentaires est donc :

$$O_r - 1 + \sum_{i=1}^{O_r}(i(2N_e + 1) - 1) \tag{5.6}$$

soit :

$$\frac{O_r(O_r - 1)(2N_e + 1)}{2} - 1 \tag{5.7}$$

Et donc, pour les N_s sorties du réseau RPN complet :

$$N_s(\frac{O_r(O_r - 1)(2N_e + 1)}{2} - 1) \tag{5.8}$$

Ce calcul ne prend pas en compte la fonction d'activation.

Le calcul de la sortie d'un HPU d'ordre O_r, si on ne prend pas en compte non plus la fonction d'activation, revient à déterminer la valeur d'un polynôme de degré O_r en un point. Les O_r puissances consécutives de x peuvent être réalisées avec $O_r - 1$ multiplications, puis ensuite il faut effectuer O_r multiplications et O_r additions. Le nombre d'opérations d'un HPU est donc :

$$3O_r - 1 \tag{5.9}$$

La tangente hyperbolique peut être calculée à l'aide d'une table de scrutation et d'un polynôme [TANG91]. Nous n'avons pas fait d'étude précise des opérations nécessaires à ce calcul, mais le nombre d'opérations est de l'ordre de quelques dizaines. Nous allons compter 50 opérations pour une tangente hyperbolique, en notant que la complexité de son calcul est généralement négligeable devant les autres opérations à effectuer, et une évaluation précise n'est pas nécessaire.

Ces calculs de complexité des différents éléments permet maintenant de déterminer celle des correcteurs eux-mêmes.

2.2.2. Complexité des correcteurs

Un récepteur OFDM classique a besoin d'une transformée de Fourier et d'un égaliseur linéaire. Le nombre d'opérations demandées est donc (équations (5.2) et (5.3)) :

$$N_p(5\log_2 N_p + 6) \tag{5.10}$$

En plus du récepteur OFDM, le correcteur RPN fréquentiel nécessite un réseau RPN d'ordre 7 à $4N_p - 2$ entrées et 2 sorties, avec une fonction d'activation qui est la somme de 3 tangentes hyperboliques (dans le cas d'une MAQ16). Ce réseau est utilisé N_p fois pour corriger les porteuses. Le nombre d'opérations requis est donc (équations (5.10) et (5.8)) :

$$N_p(5\log_2 N_p + 6) + N_p(2(\frac{7(7-1)(8N_p-3)}{2} - 1) + 2 + 3 \cdot 50) \tag{5.11}$$

soit :

$$N_p(5\log_2 N_p + 336N_p + 30) \tag{5.12}$$

Le correcteur RPN temporel ajoute au récepteur OFDM deux transformées de Fourier, un RPN d'ordre 5 à 2 entrées, 2 sorties, et une fonction d'activation en tangente hyperbolique. Le RPN et la fonction d'activation sont dans le domaine temporel et doivent être appliqués aux N_p échantillons temporels du symbole OFDM. Le nombre d'opérations requis est donc (équations (5.10), (5.2) et (5.8)) :

$$N_p(5\log_2 N_p + 6) + 10N_p\log_2 N_p + 2N_p(\frac{5(5-1)(4+1)}{2} - 1 + 50) \tag{5.13}$$

soit :

$$N_p(15\log_2 N_p + 160) \tag{5.14}$$

Pour le correcteur à HPU, en plus du récepteur OFDM et des deux transformées de Fourier, il faut déterminer le module du signal (2 multiplications, une addition, une racine carrée, donc 4 opérations), utiliser un HPU d'ordre 5 avec tangente hyperbolique, puis changer le module du signal (pour la partie réelle puis la partie imaginaire, une division par le module puis une multiplication par la sortie du HPU, donc 4 opérations au total). Le nombre d'opérations est donc de (équations (5.10), (5.2) et (5.9)) :

$$N_p(5\log_2 N_p + 6) + 10N_p\log_2 N_p + N_p(4 + 3 \cdot 5 - 1 + 50) \tag{5.15}$$

Soit :

$$N_p(15\log_2 N_p + 74) \tag{5.16}$$

Afin de déterminer de quelle manière ces complexités évoluent avec le nombre de porteuses du système OFDM, elles vont maintenant être représentées de manière graphique.

2.2.3. Comparaison des correcteurs

Nous pouvons maintenant tracer les courbes du nombre d'opérations nécessaires avec chacun des correcteurs en fonction du nombre de porteuses, dans la figure 5.6 :

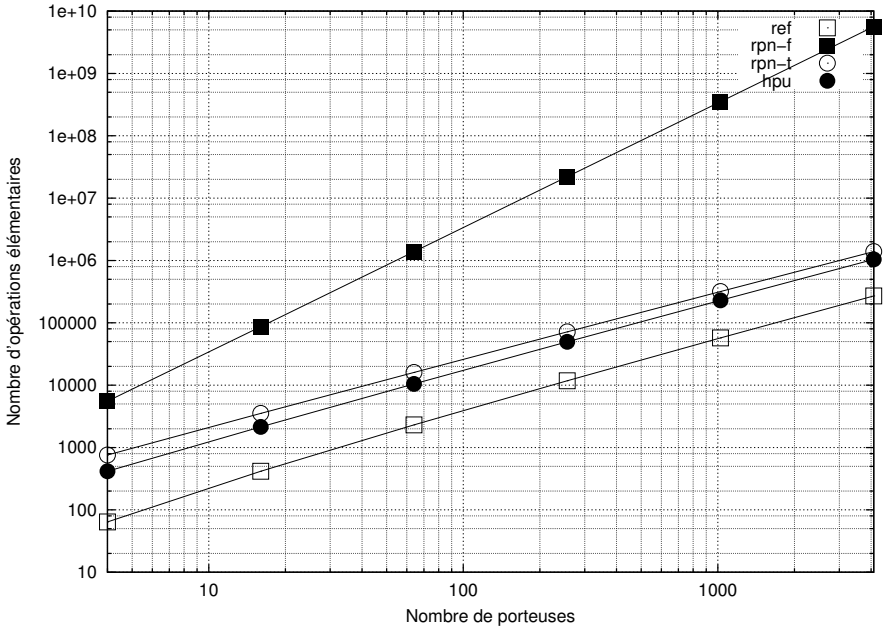

Figure 5.6. Complexité de calcul des différents correcteurs

La courbe 'ref' représente la complexité du récepteur OFDM simple sans correcteur. Contrairement à l'hypothèse de départ, même pour un faible nombre de porteuses, le correcteur RPN dans le domaine fréquentiel demande bien plus d'opérations que celui dans le domaine temporel. Les correcteurs RPN temporel et HPU demandent une puissance de calcul du même ordre de grandeur que celle requise par le récepteur OFDM, tandis que le correcteur fréquentiel en nécessite beaucoup plus. Enfin on remarque que la quantité de calcul du correcteur RPN temporel n'est pas beaucoup plus grande que celle du HPU. Les correcteurs temporels augmentent donc peu la complexité d'un récepteur OFDM.

2.3. Bilan

Le correcteur dans le domaine temporel possède beaucoup d'avantages. Il demande moins de calcul à l'utilisation, a les mêmes performances que le correcteur fréquentiel tout en étant plus simple à entraîner. Le correcteur HPU a les mêmes performances que le correcteur RPN temporel, tout en étant encore plus simple pour son apprentissage et légèrement plus rapide à l'utilisation. Par contre le réseau RPN est plus général au niveau des types de non-linéarités gérées, alors que le réseau HPU peut nécessiter une adaptation.

Le seul défaut du correcteur dans le domaine temporel est qu'il ne peut compenser efficacement les non-linéarités apporté par un limiteur. Dans ce cas, le correcteur RPN fréquentiel peut être plus utile, à condition que l'on réussisse à résoudre les problèmes de l'apprentissage. En effet, un domaine typique dans lequel ce correcteur serait adapté est l'ADSL, qui utilise 256 porteuses, et dont le défaut non linéaire est une saturation du convertisseur numérique/analogique.

Le correcteur temporel est également plus rapide à entraîner. Le tableau suivant résume les temps d'apprentissage de différents réseaux présentés au cours de ce mémoire.

Type correcteur	Type réseau	Algorithme	Temps
fréquentiel	PMC-30-25	Descente gradient	1 h 30
fréquentiel	PMC-30-25	Levenberg Marquardt	70 h
fréquentiel	RPN ordre 7	Descente gradient	1 h 50
fréquentiel	RPN ordre 7	Levenberg Marquardt	14 min
temporel	PMC-10	Levenberg Marquardt	2 min
temporel	RPN ordre 5	Levenberg Marquardt	30 s
temporel	HPU ordre 5	Levenberg Marquardt	8 s

Tableau 5.1. Temps d'apprentissage des différents réseaux présentés

3. Comparaison avec une autre approche

Après avoir comparé entre eux les différents correcteurs proposés, nous allons faire une comparaison avec une autre solution proposée dans la littérature. La plupart de ces techniques reposent sur une modification du codage canal, généralement pour abaisser le facteur de crête du signal OFDM temporel. Cependant la plupart de ces solutions ne s'appliquent qu'à un petit nombre de porteuses, ou pour des modulations à module constant, telles que les modulations de phase. Lorsque l'on augmente le nombre de porteuses ou le nombre de symboles de la constellation, le taux de codage diminue, ainsi que le débit utile, et le codage devient inutilisable en pratique [GUO02]. De plus nous avons constaté par des simulations que les correcteurs RPN ont de moins bonnes performances avec des modulations MAQ4 qu'avec des MAQ16, aussi bien en fréquentiel qu'en temporel. Ainsi comparer ces codes avec un correcteur à RPN, qui de plus n'utilise aucun codage canal, est difficile. Nous avons donc décidé de comparer notre solution à un autre système qui ne repose pas sur le codage canal.

Dans [ATAR98], l'auteur présente un correcteur placé dans le récepteur, qui simule une chaîne OFDM complète afin de retrouver le symbole qui a été émis. Son principe est donné au paragraphe 4.5 p. 37. La figure 5.7 donne les résultats obtenus sur un système OFDM à 128 porteuses, avec une modulation MAQ4 et un amplificateur non linéaire de modèle SSPA avec p=3 et un recul de -3 dB :

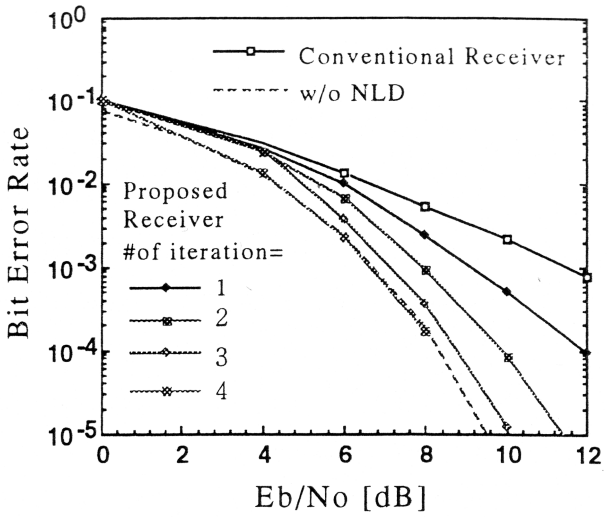

Figure 5.7. Résultats obtenus avec une postdistorsion OFDM dans un système à 128 porteuses, un amplificateur SSPA3, un recul de -3 dB et une modulation MAQ4

La courbe 'w/o NLD' est la courbe obtenue sans non-linéarité, la courbe 'Conventional Receiver' est celle obtenue sans correcteur mais avec la non-linéarité, et les 4 courbes numérotées sont celles obtenues avec la postdistorsion, au bout de 1,2,3 et 4 itérations, respectivement.

Nous avons simulé le correcteur RPN temporel dans la même situation, et les résultats sont donnés figure 5.8 :

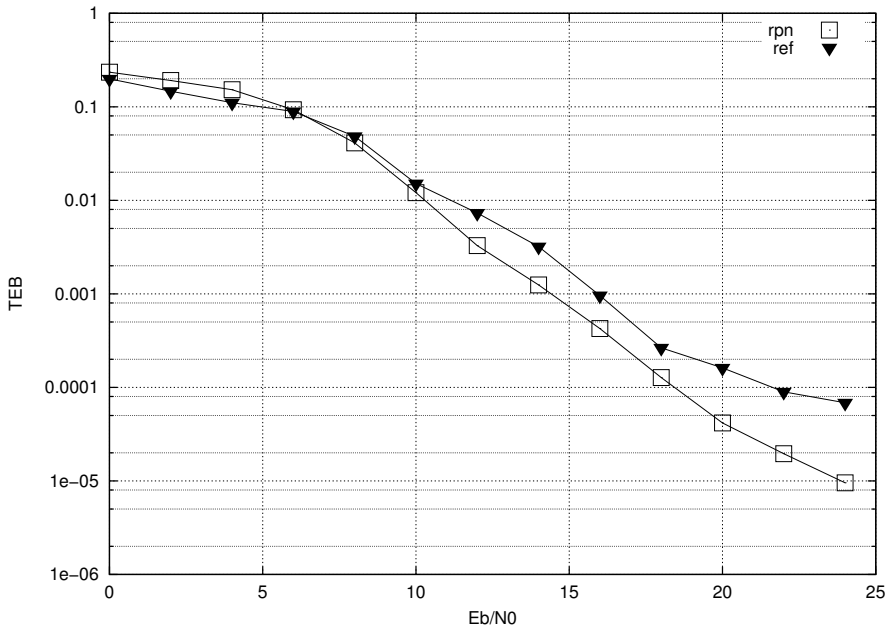

Figure 5.8. Résultats obtenus avec le correcteur RPN temporel (système OFDM avec 128 porteuses, un amplificateur SSPA, un recul de -3 dB et une modulation MAQ4)

Les différences observées entre les deux courbes de référence s'expliquent par une différence de définition du rapport signal sur bruit E_b/N_0 entre celle que nous avons utilisé et celle de l'article cité. En effet E_b représente l'énergie dépensée pour envoyer un bit d'information. Afin

de simplifier la simulation, nous avons défini cette énergie comme étant celle avant la non-linéarité. Dans notre chaîne de simulation le signal avant amplification a toujours une puissance de 1, ce qui permet de calculer simplement E_b à l'aide du débit de la source (en bits/s). Nous avons donné au modèle d'amplificateur SSPA un gain de 1 en zone linéaire, ce qui veut dire que la puissance du signal émis dans le cas linéaire est également de 1, et une simple variation de la puissance du bruit permet d'obtenir le rapport signal sur bruit voulu. Cependant dans le cas non linéaire il y a une diminution de la puissance du signal en sortie de l'amplificateur, et donc de E_b/N_0 si on le définit après la non-linéarité. Il est possible de calculer cette perte de puissance.

Comme le nombre de porteuses est assez grand, on peut approximer le signal OFDM temporel par un vecteur aléatoire gaussien. Les deux composantes gaussiennes de ce signal, les parties réelle et imaginaire, ont une puissance de $1/2$, et ainsi le signal a une puissance de 1. Le module de ce signal est une variable aléatoire de Rayleigh [GUO02] de densité de probabilité :

$$\begin{cases} p(x) = 2xe^{-x^2} & \text{si } x \geq 0 \\ p(x) = 0 & \text{si } x < 0 \end{cases} \tag{5.17}$$

Si l'on appelle X la variable aléatoire ainsi définie, la variable Y qui représente le module du signal en sortie de l'amplificateur non linéaire a pour expression :

$$\begin{aligned} &Y = h(X) \\ &\text{avec } h(x) = \frac{x}{(1 + (\frac{x}{A_0})^6)^{\frac{1}{6}}} \end{aligned} \tag{5.18}$$

L'amplitude de saturation A_0 est déterminée par le recul choisi. Ici le recul est de -3 dB et la puissance moyenne de 1, donc $A_0 = 1/\sqrt{2}$. Nous pouvons donc calculer la puissance du signal en sortie de l'amplificateur :

$$E(Y^2) = E(h(X)^2) = \int_0^{+\infty} 2x^3 \frac{e^{-x^2}}{(1 + 8x^6)^{\frac{1}{3}}}\, dx \tag{5.19}$$

Un calcul numérique de cette puissance donne environ 0,371, soit -4,3 dB. L'énergie par bit après l'amplificateur est donc celle avant l'amplificateur moins 4,3 dB. Cette différence est appliquée au rapport signal sur bruit de la courbe que nous avions obtenu, et la figure 5.9 montre le taux d'erreur binaire en fonction du nouveau rapport E_b/N_0 :

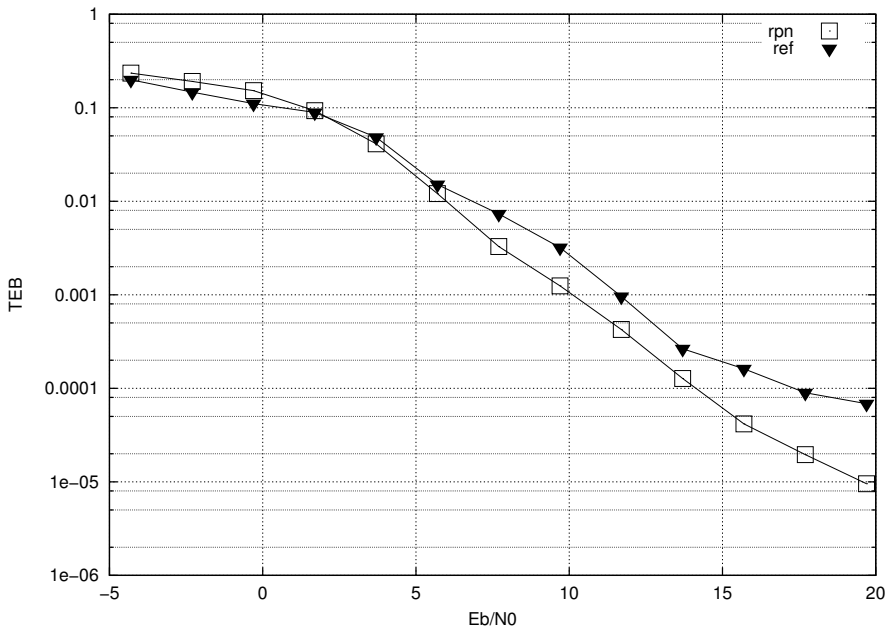

Figure 5.9. Résultats obtenus avec le correcteur RPN temporel et Eb/N0 après la nonlinéarité (système OFDM avec 128 porteuses, un amplificateur SSPA, un recul de -3 dB et une modulation MAQ4)

La courbe de référence est maintenant comparable à celle de l'article cité. Les petites différences qui subsistent s'expliquent probablement par des choix différents de certains paramètres de simulation, tels que le suréchantillonnage pour simuler le canal, ainsi que les caractéristiques du filtre de réception qui doit rejeter les intermodulations crées par la non-linéarité.

Pour un taux d'erreur binaire de 10^{-2} le correcteur à RPN temporel apporte un gain de 1 dB environ, et presque 2 dB pour un taux d'erreur binaire de 10^{-3}. Avec un rapport E_b/N_0 de 20 dB ce taux d'erreur binaire est divisé par 6. Si l'on compare avec les résultats présentés dans l'article [ATAR98] dont les résultats apparaissent figure 5.7, le correcteur à RPN a des performances similaires à la postdistorsion OFDM avec 2 itérations pour de faibles rapports signal sur bruit (inférieur à 6 dB) et a de bien moins bonnes performances pour des rapports signal sur bruit supérieurs. Par contre ce correcteur demande beaucoup moins de calcul.

En effet, au cours d'une itération, l'algorithme de postdistorsion simule la chaîne OFDM en retransmettant le symbole reçu, ainsi que l'ensemble des symboles obtenus en changeant un seul bit au symbole reçu. Le symbole obtenu en sortie de la simulation est comparé au symbole réellement reçu, et celui qui est le plus proche est retenu comme nouveau symbole reçu. Ce nouveau symbole est soit définitivement retenu, soit sert à une nouvelle itération. Donc pour chaque symbole reçu, il faut faire $2N_p + 1$ simulations de la chaîne OFDM (dans le cas d'une MAQ4, où chaque porteuse code 2 bits). Si l'on néglige le nombre d'opérations nécessaires à la simulation de l'amplificateur non linéaire, à la conversion binaire à symbole et à la comparaison entre symboles, il reste celui nécessaire aux deux transformées de Fourrier. Ainsi le nombre d'opérations requis pour une itération est de (équation (5.2)) :

$$(2N_p + 1)5N_p \log_2 N_p \tag{5.20}$$

La figure 5.10 montre l'évolution du nombre d'opérations requis par la post-distorsion OFDM (avec 1,2 et 4 itérations, respectivement postd-1, postd-2 et postd-4) et le correcteur à RPN temporel (rpn-t) en fonction du nombre de porteuses :

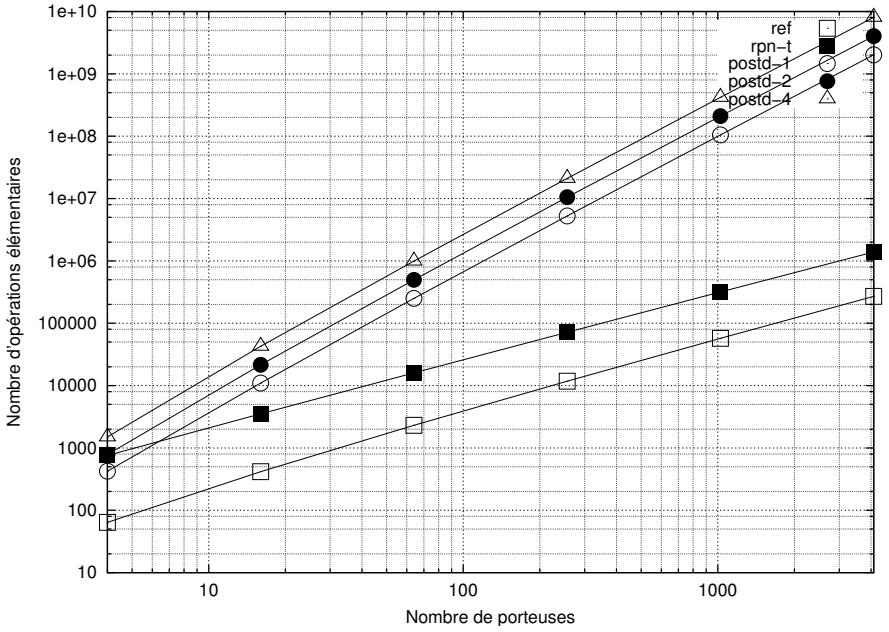

Figure 5.10. Nombre d'opérations nécessaire à l'exécution des différents algorithmes de postdistorsion en fonction du nombre de porteuses

La puissance de calcul demandée augmente dans de très grandes proportions avec le nombre de porteuses. Pour 100 porteuses, la postdistorsion avec une itération demande déjà plus de 15 fois plus de calculs que le récepteur OFDM, et 60 fois plus que le récepteur OFDM classique. A 1000 porteuses, ces facteurs passent respectivement à 330 et 2000.

Ainsi, la postdistorsion permet de meilleures performances que le correcteur à RPN, mais au prix d'une plus grande complexité de calcul. Si le nombre de porteuses est important ou la puissance de calcul limitée, le correcteur à RPN possède l'avantage d'être plus simple. De plus la méthode de postdistorsion nécessite de connaître parfaitement le modèle de l'amplificateur et du canal afin d'effectuer correctement la simulation. Enfin les performances de la postdistorsion n'ont pas été évaluées dans le cas d'une modulation à plus grand nombre d'états, tels que la MAQ16. Comme on peut le voir figure 5.11, avec cette modulation le correcteur à RPN offre un gain bien supérieur à celui obtenu avec une modulation MAQ4. De plus la complexité du correcteur RPN, aussi bien lors de l'apprentissage que lors de la simulation est la même avec les deux modulations. Avec la méthode de la postdistorsion, le nombre de simulations à effectuer doit être doublé dans ce cas.

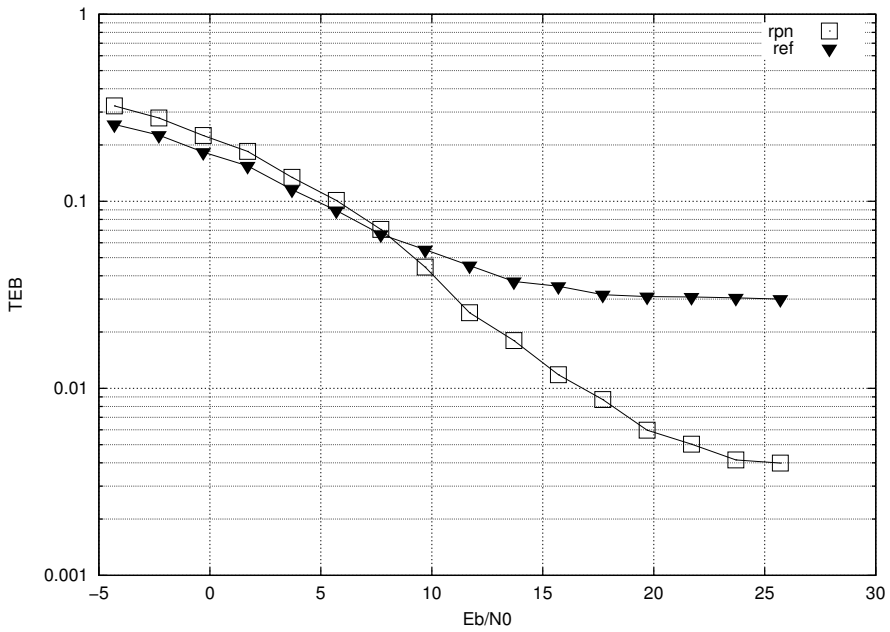

Figure 5.11. Résultats obtenus avec le correcteur RPN temporel et Eb/N0 après la non-linéarité (système OFDM avec 128 porteuses, un amplificateur SSPA, un recul de -3 dB et une modulation MAQ16)

Conclusion

Dans ce chapitre nous avons abordé la mise en pratique des correcteurs à réseaux de neurones dans un système OFDM, tout d'abord en constatant que le correcteur temporel est relativement indépendant du canal, et donc qu'il peut être mis en oeuvre dans un canal multitrajets, puis en évoquant la question de l'apprentissage.

Dans les chapitres 3 et 4 les deux correcteurs ont été simulés avec différentes formes de non linéarités. Le correcteur fréquentiel a montré de bons résultats dans tous les cas. Par contre le correcteur temporel n'a pas apporté de gain important avec un limiteur. En effet cette non-linéarité, que l'on retrouve dans les liaisons filaires comme l'ADSL ou les liaisons radio avec prédistorsion, n'est pas inversible en prenant chaque symbole temporel individuellement. Le correcteur fréquentiel, en utilisant l'information disponible sur toutes les porteuses pour chaque symbole OFDM, est lui capable de retrouver l'information originale avec une meilleure probabilité. Pour les non-linéarités de type SSPA les deux correcteurs ont montré de bonnes performances, et une adaptabilité aux changements de modèles. Enfin pour les non-linéarités de type Tubes à Ondes Progressives, qui font également intervenir une non-linéarité de phase, aucun essai n'a été fait durant cette thèse, mais il est probable que les réseaux RPN pourront d'adapter à cette autre forme de non linéarité. Enfin le correcteur simplifié HPU a les mêmes performances que le RPN mais en gérant des formes plus spécifiques de non-linéarités. En particulier dans le cas des Tubes à Ondes Progressives il serait nécessaire de modifier le correcteur en ajoutant un second HPU qui puisse corriger la non linéarité de phase.

Dans ce chapitre il a été montré que les correcteurs temporels et fréquentiels ont les mêmes performances sur un système à 4 porteuses et un amplificateur de type SSPA. Cependant le correcteur temporel nécessite une puissance de calcul bien inférieure à celle du correcteur fréquentiel, et qui reste du même ordre que celle d'un récepteur OFDM classique. Comme de plus le correcteur temporel est le seul qui conserve des performances intéressantes en

augmentant le nombre de porteuses, c'est ce dernier qui est le plus applicable à des systèmes OFDM courants.

Le correcteur à RPN temporel a enfin été comparé avec un autre correcteur que l'on peut trouver dans la littérature. Il a été simulé dans les mêmes conditions, et les résultats montrent des performances similaires pour de faibles rapports signal sur bruit (en dessous de 6 dB). Lorsque la niveau de bruit diminue le correcteur à RPN est bien moins performant. Cependant il a l'avantage de demander beaucoup moins de puissance de calcul pour le récepteur, et il s'adapte à des modulations ayant plus de symboles sans augmentation de complexité. Les deux correcteurs sont donc adaptés à des conditions différentes.

Conclusion et Perspectives

Ce mémoire présente des travaux effectués au sein de l'équipe ETSN de Supélec, campus de Rennes, sur la réduction des effets des non linéarités dans une modulation OFDM. Nous avons en particulier cherché à réduire le taux d'erreur binaire d'une transmission OFDM en ajoutant un réseau de neurones au sein du récepteur, dont le rôle est de compenser les distorsions introduites par l'amplificateur non linéaire. Des correcteurs utilisant diverses architectures neuronales ont été présentés avec les résultats obtenus. Nous nous sommes de plus intéressés à leur mise en oeuvre, et en particulier à la puissance de calcul nécessaire ainsi que leur adaptation au canaux multitrajets.

Le chapitre 1 présente les modulations multiporteuses. Aujourd'hui, plusieurs standards reposent sur la modulation OFDM, en particulier en raison de la simplicité de l'égalisation du canal. Ceci permet de transmettre avec plus d'efficacité des données sur des canaux multitrajets (tels que les communications à l'intérieur d'un bâtiment). Cependant le signal OFDM temporel possède un facteur de crête élevé, et pour des raisons de coût et d'efficacité énergétique, il n'est généralement pas possible de placer dans l'émetteur un amplificateur qui possède une réponse linéaire pour l'intégralité de la dynamique du signal. Ainsi, des erreurs dues à la non-linéarité de l'amplificateur apparaissent dans la transmission. Aujourd'hui, diverses techniques sont étudiées pour diminuer ces effets, en particulier en intervenant sur le codage canal.

Le chapitre 2 est une introduction aux réseaux de neurones en approximation de fonctions et insiste en particulier sur leur mise en oeuvre. Après en avoir posé la problématique, on présente les deux familles de réseaux de neurones les plus classiques, les PMC et les GRBF. Les PMC sont des réseaux basés sur des produits scalaires, qui réalisent généralement des approximations plutôt globales de fonctions, c'est à dire sur de grands intervalles, tandis que les GRBF sont basés sur les distances, et produisent des approximations plutôt locales. A l'aide d'une base d'apprentissage constituée d'un certain nombre de points de la fonction à approximer, différents algorithmes permettent de trouver un jeu de paramètres adaptés, cette étape étant appelée apprentissage du réseau. Enfin les réseaux d'ordre supérieur sont présentés. Ces réseaux peuvent exploiter les corrélations d'ordre supérieur entre les entrées afin de réaliser des fonctions plus complexes, tout en nécessitant moins de paramètres que les PMC ou les GRBF sur certaines classes de problèmes. On insiste en particulier sur les réseaux pi-sigma, ainsi que sur leur généralisation, les RPN. L'algorithme d'apprentissage est proche de celui du perceptron, mais nécessite quelques adaptations. Ces deux premiers chapitres ont permis d'introduire les différents concepts qui seront employés dans le reste de ce mémoire de thèse.

Dans le chapitre 3, le premier correcteur étudié est présenté. Celui-ci est placé dans le récepteur, après l'égalisation du canal, dans le domaine fréquentiel. Une étude de la non-linéarité dans le domaine fréquentiel permet de déduire que les termes d'intermodulation peuvent s'exprimer sous la forme de somme de produits entre les symboles émis sur les différentes porteuses. On établit donc que le réseau de neurones doit avoir en entrée le symbole OFDM complet, même pour corriger uniquement le symbole sur une porteuse donnée. Cette étude établit de plus des règles de symétrie dans le domaine fréquentiel de la non-linéarité, et il est ainsi possible de simplifier le réseau de neurones. Le premier réseau de neurones employé pour réaliser ce correcteur est un PMC. Malgré de nombreuses tentatives sur plusieurs systèmes OFDM avec différents paramètres de simulation et d'apprentissage, il n'a pas été possible d'obtenir de

résultats satisfaisants. En effet l'algorithme d'apprentissage ne parvient pas à faire converger le réseau vers une solution qui réduirait le taux d'erreur binaire. Le second correcteur présenté est basé sur un RPN, un réseau de neurones d'ordre supérieur. Les performances sont intéressantes sur 4 porteuses avec une modulation MAQ16, puisque l'on constate un gain de 1,5 dB pour un taux d'erreur binaire de 10^{-2} dans un système avec un modèle d'amplificateur non linéaire SSPA, et un gain de 5 dB au même taux d'erreur binaire dans un système avec limiteur. Par contre les performances sont moins intéressantes avec 8 porteuses, et avec 16 porteuses le correcteur n'est plus efficace. Le problème de l'apprentissage se pose à nouveau avec le correcteur à RPN, mais pour un nombre de porteuses plus élevé qu'avec le correcteur à PMC.

Pour réaliser un système adapté à un plus grand nombre de porteuses, l'idée proposée dans le chapitre 4 est de placer le réseau de neurones dans le domaine temporel. Ce nouveau correcteur est plus proche de solutions existantes avec des modulations monoporteuses, tout en ayant quelques caractéristiques propres. Tout d'abord l'amplificateur sature bien plus avec un signal OFDM temporel, et donc la non-linéarité est plus prononcée, et de plus le réseau de neurones doit réaliser une approximation de fonction sur un intervalle, tandis que dans le cas d'un signal monoporteuse, il doit faire une classification parmi un nombre fini de symboles placés sur une constellation. Enfin il n'est pas possible d'inverser parfaitement la non linéarité de l'amplificateur car une partie du signal temporel, présente en dehors de la bande utile, est fortement atténuée au cours de la transmission. Dans le principe proposé, le signal OFDM reçu est tout d'abord transformé dans le domaine fréquentiel afin d'effectuer l'égalisation du canal, puis ramené à nouveau dans le domaine temporel. Ensuite le réseau de neurones effectue la compensation de la non-linéarité, puis le signal est une fois encore transformé dans le domaine fréquentiel afin d'extraire le symbole OFDM reçu. Deux correcteurs sont présentés dans ce chapitre, avec les réseaux de neurones PMC et RPN. Les deux réseaux permettent de réduire le nombre d'erreurs, mais c'est également le RPN qui obtient les meilleures performances, avec un gain mesuré de 6 à 8 dB pour un taux d'erreur binaire de 10^{-2} dans un système OFDM à 48 porteuses, une modulation MAQ16 et des modèles d'amplificateurs SSPA. Le système présenté permet donc dans ces conditions de diviser la puissance de l'amplificateur, et donc sa consommation d'énergie, par au moins 4 tout en conservant la même qualité de transmission. Par contre dans le cas du limiteur la compensation des erreurs est moins efficace.

La chapitre 5 replace les travaux effectués dans le contexte des applications multiporteuses, en s'intéressant tout d'abord à quelques aspects de leur mise en pratique et à leurs performances en présence d'un canal multitrajet. Deux possibilités sont envisagées pour l'apprentissage du réseau. Premièrement celui-ci peut-être effectué une seule fois par amplificateur, et les paramètres obtenus peuvent être mémorisés et transmis au récepteur au début de la transmission. Une autre solution est qu'il soit effectué par le récepteur à chaque fois qu'une transmission est établie, à l'aide de symboles OFDM de calibration connus de l'émetteur et du récepteur. Dans ce dernier cas il faut remarquer que pour obtenir des performances optimales, il est nécessaire de déterminer expérimentalement le meilleur rapport signal sur bruit du canal lors de l'apprentissage. Des simulations ont également été effectuées avec un canal multitrajet, et celles-ci montrent que le correcteur à RPN temporel garde ses bonnes performances, ce qui était attendu étant donné que l'égalisation du canal et celle de la non-linéarité sont indépendantes dans ce correcteur. On mesure en effet un gain de près de 10 dB pour un taux d'erreur binaire de 10^{-2} dans un canal multitrajet avec une modulation OFDM composée de 512 porteuses. Ensuite dans une seconde partie les correcteurs présentés dans les chapitres 3 et 4 sont comparés. Les deux types de correcteurs sont capables de s'adapter à différentes non-linéarités, sauf dans le cas du limiteur, dans lequel le correcteur temporel est

bien moins efficace. Dans un système à 4 porteuses, les deux correcteurs ont des performances très similaires, mais lorsque l'on augmente le nombre de porteuses, seul le correcteur temporel garde de bonnes performances. De plus ce dernier demande beaucoup moins de calcul que le correcteur fréquentiel. En effet dans un système à 1000 porteuses le système OFDM avec correcteur temporel a besoin d'environ 8 fois plus de puissance de calcul que le récepteur OFDM simple, tandis que le correcteur fréquentiel en nécessiterait 10000 fois plus. Enfin le correcteur RPN temporel est comparé à un autre type de correcteur, dit postdistorsion OFDM et présenté dans [NISH96]. Il s'avère que le correcteur RPN temporel présenté dans le chapitre 4 est bien moins complexe que la postdistorsion et demande beaucoup moins de calculs, surtout avec un nombre de porteuses élevé. Ceci se fait au détriment d'une légère dégradation des performances, principalement avec un rapport signal sur bruit élevé. Dans les conditions présentées dans l'article, le système cité permet un gain de 2 dB pour un taux d'erreur binaire de 10^{-2}, tandis qu'avec le correcteur à RPN temporel dans les mêmes conditions, on mesure un gain de 1 dB. Par contre ce dernier demande environ 100 fois moins de calcul que le système cité.

Les travaux accomplis durant cette thèse ouvrent plusieurs perspectives de travaux futurs. Tout d'abord il est possible de prolonger le travail présenté dans le chapitre 3. En effet l'apprentissage du réseau est difficile lorsque le nombre de porteuses est supérieur à 8. Il est possible de travailler soit sur l'algorithme d'apprentissage lui-même, soit en utilisant une autre technique d'optimisation, soit en le guidant à l'aide des symétries de la non-linéarité dans le domaine fréquentiel. Cette symétrie pourrait aussi permettre de simplifier l'architecture du réseau de neurones, facilitant ainsi son apprentissage. Il est également possible d'implémenter d'autres types de réseaux de neurones, tels que les SVM, qui n'ont pas été étudiés ici.

Le travail présenté dans le chapitre 4 peut aussi être prolongé selon différents axes. Même si nous avons précisé que le problème de l'inversion de la non-linéarité n'a pas exactement les mêmes caractéristiques que celui avec une modulation monoporteuse, les simulations ont montré que le correcteur à RPN avait de bien meilleures performances que celui avec un PMC. Une application des réseaux d'ordre supérieur aux modulations plus classiques serait intéressante à étudier et pourrait apporter de meilleures performances que les correcteurs neuronaux déjà étudiés dans la littérature. De plus, l'égalisation du canal et la correction de la non-linéarité sont totalement indépendantes dans le système que nous proposons. Comme montré dans le chapitre 5, l'apprentissage successif des deux modules ne pose pas de problèmes dans un canal multitrajets, mais il est tout de même possible que l'égaliseur soit perturbé par la non-linéarité. Il serait peut-être intéressant d'étudier un apprentissage simultané des deux modules afin de l'accélérer, et également de pouvoir réduire une éventuelle perte de performance causée par l'égaliseur de canal.

Les deux correcteurs n'ont pas été étudiés en présence d'un code correcteur d'erreurs. La distribution des erreurs à l'issu de ces correcteurs n'a pas été déterminée, et une étude en présence d'un code canal est nécessaire avant de projeter une implémentation sur un système existant. Si les réseaux neuronaux ne modifient pas la distribution des erreurs, tout code utilisé en OFDM pourra être employé.

Le travail accompli dans cette thèse peut également s'inscrire dans un autre domaine de recherche de l'équipe ETSN, la radio logicielle. Son objectif est de réaliser des systèmes qui sont capables de s'adapter à différents protocoles de communication. Ces modifications peuvent être effectuées aussi bien au niveau du matériel que du logiciel, à l'aide de processeurs et composants reconfigurables dits "system on chip". Lors d'un changement de protocole,

il est probable que l'amplificateur d'émission soit également différent, ou utilisé à un point de fonctionnement différent. Les caractéristiques de sa non-linéarité peuvent également se modifier, et donc un correcteur capable de s'adapter à différentes non-linéarités peut s'avérer utile. L'intégration d'un des correcteurs présenté dans ce mémoire au sein d'un système de radio logicielle, ainsi que son interaction avec les différents protocoles de communication peut constituer un dernier axe de recherche prolongeant les travaux déjà effectués.

Annexe A. Quelques algorithmes d'optimisation

Dans cette annexe, deux algorithmes d'optimisation qui ont servi durant les travaux présentés dans ce mémoire sont présentés. Le problème résolu par ces algorithmes est le suivant : soit f une fonction, dépendant d'un vecteur de paramètres $\mathbf{p}$. On ne connaît pas l'expression de f, mais on sait calculer numériquement quelques caractéristiques de cette fonction (valeur ou dérivée en un point par exemple). On cherche, numériquement également, le vecteur $\mathbf{p}$ qui minimise la quantité $f(\mathbf{p})$.

Dans le contexte des réseaux de neurones, f est la fonction de performance, généralement l'erreur quadratique moyenne calculée sur une base d'apprentissage, et $\mathbf{p}$ est un vecteur regroupant tous les poids du réseau. Un algorithme d'optimisation permet d'effectuer un apprentissage du réseau, c'est à dire de trouver les poids qui minimisent l'erreur quadratique moyenne.

Les deux algorithmes ci-après ne trouveront qu'un minimum local de la fonction. Si f possède plusieurs minima locaux, celui vers lequel convergera l'algorithme dépend du point de départ choisi et des paramètres de l'algorithme. On ne peut garantir de converger vers un minimum global.

1. Descente de gradient

L'algorithme d'optimisation le plus simple est la descente de gradient, dont le principe est de partir d'un point aléatoire puis de se déplacer dans la direction de la plus forte pente. En appliquant un certain nombre d'itérations, l'algorithme converge vers une solution qui est un minimum local de f.

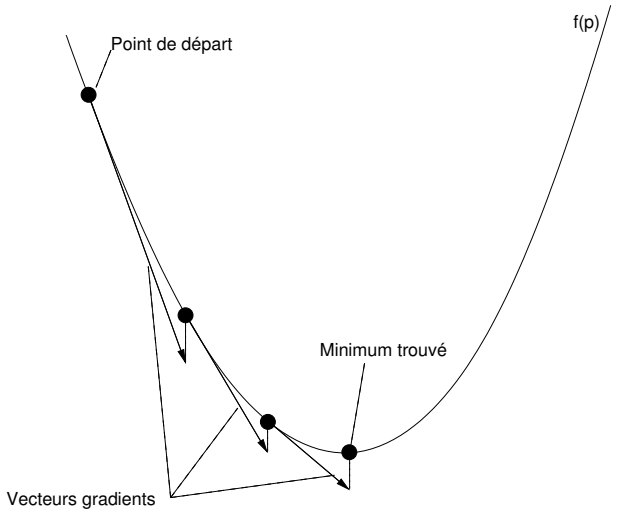

Figure A.1. Illustration du principe de la descente de gradient, dans le cas d'une fonction à une variable

On commence donc par choisir un vecteur $\mathbf{p}_0$ de manière aléatoire. Puis pour l'itération numéro i on calcule le gradient de f au point $\mathbf{p}_{i-1}$:

$$\mathbf{g}_i = \nabla f(\mathbf{p}_{i-1}) \tag{A.1}$$

Le nouveau vecteur de paramètres calculé est :

$$\mathbf{p}_i = \mathbf{p}_{i-1} - \eta \mathbf{g}_i \tag{A.2}$$

où η est une constante qui ajuste la vitesse de convergence de l'algorithme, déterminée empiriquement. Une fois le nouveau vecteur $\mathbf{p}_i$ calculé on passe à l'itération suivante. Si η est trop grande, l'algorithme n'est pas stable et oscille autour d'une solution, et si η est trop petite, un très grand nombre d'itérations sera nécessaire pour converger vers la solution, et la probabilité de convergence vers une solution locale est plus grande. Plusieurs critères peuvent être définis pour arrêter l'algorithme : on peut limiter à un certain nombre d'itérations, ou arrêter lorsque $f(\mathbf{p}_i)$ atteint un certain seuil minimal ou encore lorsque le vecteur évolue peu, c'est à dire quand la valeur suivante atteint un seuil minimal :

$$\frac{\|\mathbf{g}_i\|}{\|\mathbf{p}_i\|} \tag{A.3}$$

Ce dernier critère peut présenter le défaut d'arrêter l'algorithme trop tôt si la fonction présente des plateaux. Le choix du meilleur critère ainsi que le seuil à fixer est généralement trouvé de manière empirique. Il est également possible de prendre une combinaison de ces différents critères.

Le choix du coefficient η peut être délicat dans certains cas. Par exemple si f possède par endroits de grands plateaux, il faudrait avoir un coefficient η grand pour pouvoir s'en affranchir avec peu d'itérations. Si en d'autres endroits f évolue au contraire très rapidement, il faut qu'il soit faible pour que l'algorithme soit stable. Une variante peut être utile dans ce cas, la descente de gradient adaptative.

Dans une descente de gradient adaptative, le coefficient η est également ajusté à chaque itération, suivant l'évolution de la valeur de $f(\mathbf{p}_i)$. Si $f(\mathbf{p}_i)$ diminue, il est probable que l'on pourrait aller plus vite en augmentant légèrement η, et au contraire si $f(\mathbf{p}_i)$ augmente, cela veut dire que le coefficient η est trop grand et qu'il faut le diminuer. Donc on décide d'augmenter η (de 10% par exemple) si $f(\mathbf{p}_i)$ diminue, et de le réduire (en le divisant par 2 par exemple) si $f(\mathbf{p}_i)$ augmente. Cette approche permet généralement de réduire le nombre d'itérations requis, et s'est révélée efficace avec tous les réseaux de neurones que nous avons testés.

La descente de gradient peut être appliquée de deux manières lorsque l'on évalue la fonction à l'aide d'une base d'exemples. La méthode que nous avons employé, et décrite ci-dessus, est celle du gradient total. Le vecteur $\mathbf{g_i}$ est calculé avec tous les exemples de la base d'apprentissage à chaque itération, et le nouveau vecteur de paramètres est déterminé après avoir parcouru toute la base. Dans une autre méthode, dite du gradient stochastique, le vecteur $\mathbf{g_i}$ est calculé avec chaque exemple, et le vecteur de paramètres est recalculé entre chaque exemple. Cette dernière méthode est particulièrement adaptée aux systèmes dits online, pour lesquels les exemples sont communiqués l'un après l'autre pendant l'optimisation, alors que pour la méthode du gradient total il est nécessaire d'avoir la base complète avant de commencer la première itération.

2. Algorithme de Levenberg Marquardt

Une autre manière de diminuer le nombre d'itérations d'un algorithme d'optimisation est d'utiliser les dérivées secondes de f. En effet le gradient donne une direction vers laquelle se déplacer pour trouver le minimum, mais ne donne pas le pas. Dans la descente de gradient classique ce pas est un coefficient fixe, et dans la variante adaptative il peut varier à chaque itération. Mais la dérivée seconde de f est liée au rayon de courbure de la fonction, et permet donc de déterminer ce pas de manière plus fine.

En effet si l'on suppose que f est une fonction quadratique :

$$f(\mathbf{p}) = a + \mathbf{b}^T\mathbf{p} + \mathbf{p}^T\mathbf{C}\mathbf{p} \tag{A.4}$$

où $\mathbf{x}^T$ est la transposée du vecteur $\mathbf{x}$ et $\mathbf{C}$ est une matrice symétrique, on peut trouver l'extremum de la fonction, il s'agit du point auquel la dérivée de f s'annule :

$$\nabla f = 0 \;\Leftrightarrow\; \mathbf{b} + 2\mathbf{C}\mathbf{p} = 0 \tag{A.5}$$

Soit :

$$\mathbf{p} = -2\mathbf{C}^{-1}\mathbf{b} \tag{A.6}$$

A condition que $\mathbf{C}$ soit inversible. Pour une fonction f quelconque, il est possible de l'approximer localement en un point $\mathbf{p}_i$ par une fonction quadratique, en utilisant ses dérivées première et seconde, et avec l'équation (A.6) déterminer le vecteur pour l'itération suivante d'un algorithme d'optimisation plus évolué que la descente de gradient. Mais le calcul des dérivées secondes peut être très long, tout d'abord parce que le nombre de dérivées secondes est le carré de celui des dérivées premières, et également parce que la dérivée seconde de f peut être assez complexe. De nombreux algorithmes, peut-être abusivement appelés algorithmes d'ordre 2, utilisent en fait une approximation des dérivées secondes calculées à partir de dérivées premières. Cependant ils gardent l'avantage de nécessiter beaucoup moins d'itérations qu'une descente de gradient.

L'algorithme de Levenberg Marquardt [MARQ63] fait partie de ces algorithmes, et s'applique au cas particulier où f est une erreur quadratique moyenne. On peut donc l'exprimer sous la forme :

$$f(\mathbf{p}) = \langle (g(\mathbf{x}, \mathbf{p}) - y)^2 \rangle \tag{A.7}$$

où g désigne une fonction de deux vecteurs $\mathbf{x}$ et $\mathbf{p}$ et $\langle \cdot \rangle$ désigne la moyenne calculée sur un ensemble de couples $(\mathbf{x}, y)$. L'on se place dans le cas où g est une fonction scalaire afin de simplifier la notation, mais la même démarche peut être faite si $\mathbf{g}$ est une fonction vectorielle. Dans la suite de cette section toutes les dérivées sont en fonction du vecteur $\mathbf{p}$. C'est en effet uniquement ce vecteur que l'on fait varier afin de trouver le minimum de f.

On suppose, comme pour la descente de gradient, que l'on se trouve à une itération numéro i, et que l'on cherche à calculer un nouveau vecteur $\mathbf{p}_i$ en fonction de $\mathbf{p}_{i-1}$, tel que $f(\mathbf{p}_i)$ se rapproche plus d'un minimum local de f. Pour cela on calcule une approximation quadratique

$\hat{f}$ de f à partir d'une approximation linéaire $\hat{g}$ de g autour du point $\mathbf{p}_{i-1}$. En déterminant le point $\mathbf{p}$ auquel le gradient de $\hat{f}$ s'annule, on obtient :

$$\mathbf{p} = \mathbf{p}_{i-1} - \mathbf{H}^{-1}\mathbf{d} \tag{A.8}$$

avec :

$$\begin{array}{l} \mathbf{d} = \langle (g(\mathbf{x}, \mathbf{p}_{i-1}) - y) \nabla g(\mathbf{x}, \mathbf{p}_{i-1}) \rangle \\ \mathbf{H} = \langle \nabla g(\mathbf{x}, \mathbf{p}_{i-1}) \nabla g(\mathbf{x}, \mathbf{p}_{i-1})^T \rangle \end{array} \tag{A.9}$$

à condition que $\mathbf{H}$ soit inversible. La matrice $\mathbf{H}$ est une approximation du Hessien de f, calculée à partir du gradient de g. L'équation précédente pourrait servir dans un algorithme d'optimisation, qui permet de calculer $\mathbf{p}_i$ à partir de $\mathbf{p}_{i-1}$ au cours de l'itération i. Mais ceci n'est efficace en pratique que si g est effectivement proche d'une droite autour du point $\mathbf{p}_{i-1}$. Dans le cas contraire cet algorithme donne de très mauvais résultats.

L'idée de Levenberg est donc d'utiliser cette approche quadratique dans les zones où g est quasi-linéaire, et une descente de gradient dans les autres cas. Le pas d'une itération de cet algorithme est calculé de la manière suivante :

$$\mathbf{p_i} = \mathbf{p}_{i-1} - (\mathbf{H} + \lambda \mathbf{I})^{-1}\mathbf{d} \tag{A.10}$$

Lorsque λ est faible, cette équation est équivalente à (A.8), et le nouveau vecteur de paramètres est déterminé avec l'approximation quadratique de f. Lorsque λ est grand, cette équation est équivalente à :

$$\begin{array}{rcl} \mathbf{p_i} & = & \mathbf{p}_{i-1} - \dfrac{1}{\lambda}\mathbf{d} \\ & = & \mathbf{p}_{i-1} - \dfrac{1}{\lambda}\langle (g(\mathbf{x}, \mathbf{p}_{i-1}) - y) \nabla g(\mathbf{x}, \mathbf{p}_{i-1}) \rangle \\ & = & \mathbf{p}_{i-1} - \dfrac{1}{2\lambda}\nabla f(\mathbf{x}, \mathbf{p}_{i-1}) \end{array} \tag{A.11}$$

Ce qui correspond bien à une descente de gradient. Pour des valeurs intermédiaires de λ l'algorithme est un mélange entre la descente de gradient et l'approche quadratique basée sur l'approximation linéaire de g. Ce coefficient λ est modifié à chaque itération, comme pour la descente de gradient adaptative. Si $f(\mathbf{p}_i)$ diminue au cours de l'itération, on diminue λ (en le divisant par 10 par exemple), et l'on se rapproche ainsi de la méthode quadratique. Au contraire si $f(\mathbf{p}_i)$ augmente, cela signifie que nous nous trouvons dans une région dans laquelle g n'est pas très linéaire, et donc on augmente λ (en le multipliant par 10 par exemple) afin de se rapprocher de la descente de gradient.

Cet algorithme a ensuite été amélioré par Marquardt, le pas de l'itération étant défini cette fois par :

$$\mathbf{p_i} = \mathbf{p}_{i-1} - (\mathbf{H} + \lambda diag(\mathbf{H}))^{-1}\mathbf{d} \tag{A.12}$$

La matrice identité a été remplacée par la diagonale de $\mathbf{H}$. Le but est ici de modifier le comportement de l'algorithme dans les cas où λ est grand, c'est à dire lorsque l'on est proche d'une descente de gradient. Avec cette modification l'on se déplace plus vite dans les directions vers lesquelles le gradient est plus fiable, afin d'éviter de passer de nombreuses itérations sur un plateau. Ceci est appelé l'algorithme de Levenberg Marquardt.

En pratique cet algorithme, en particulier dans le cas des réseaux de neurones, permet de converger avec beaucoup moins d'itérations. Mais chaque itération demande plus de calculs, en particulier pour l'inversion de la matrice $\mathbf{H}$, et son utilisation se limite donc aux cas où le nombre de paramètres à optimiser n'est pas très élevé. En effet le nombre d'opérations nécessaires à l'inversion d'une matrice est proportionnel à N^3, N étant la taille de la matrice, et ici également la taille du vecteur $\mathbf{p}$.

Index des notations

δ_{ij} : symbole de Kronecker (p. 17).

Δf : largeur de bande du signal à moduler (p. 14).

ϵ : performance d'un système d'approximation de fonction. Généralement une erreur quadratique moyenne (p. 42).

ϵ_a : performance ϵ d'un système d'approximation de fonction, calculée sur la base d'apprentissage (p. 43).

ϵ_v : performance ϵ d'un système d'approximation de fonction, calculée sur la base de validation (p. 43).

η : coefficient régulant la vitesse de convergence d'un algorithme de descente du gradient (p. 138).

σ_j : coefficient d'étalement du neurone j de la première couche d'un réseau GRBF (p. 49).

τ_R : étalement des retards d'un canal multitrajets (p. 19).

$\phi(r)$: conversion AM/PM dans un modèle d'amplificateur non linéaire (p. 66).

$\psi_k(t)$: élément de la base de signaux élémentaires pour une modulation monoporteuse (p. 15).

$\psi_{j,k}$: élément de la base de signaux élémentaires pour une modulation multiporteuses. k est l'indice de la porteuse et j celui du symbole OFDM (p. 22).

$a(r)$: conversion AM/AM dans un modèle d'amplificateur non linéaire (p. 66).

$a(t)$: signal réel à moduler en amplitude en modulation monoporteuse (p. 14).

a_k : partie réelle de c_k (p. 13).

a_l : coefficient du développement en série entière de $a(r)$ (p. 67).

$\mathbf{a_l}$: vecteur contenant les sorties de tous les neurones de la couche l d'un perceptron multicouches. Correspond donc également aux entrées de la couche $l+1$. Par extension $\mathbf{a_0}$ est l'entrée du réseau (p. 54).

B_c : bande de cohérence d'un canal de propagation (p. 20).

b_k : partie imaginaire de c_k (p. 13).

b_i : biais du neurone i de la seconde couche d'un réseau GRBF (p. 49), ou du neurone i de la première couche d'un réseau pi-sigma (p. 61).

$b_{i,j}$: biais du neurone i de la première couche du réseau pi-sigma d'ordre j composant un réseau RPN (p. 63).

$\mathbf{b_l}$: vecteur contenant tous les biais des neurones de la couche l d'un perceptron multicouches (p. 54).

$b_{l,i}$: biais du neurone i de la couche l d'un PMC (p. 54).

$\mathbf{c_j}$: symbole OFDM à transmettre, j étant le numéro du symbole. Vecteur de composantes $c_{j,k}$, où k est le nombre de porteuses (p. 23).

c_k : symbole complexe représentant plusieurs bits à transmettre par une modulation d'amplitude en quadrature. k est le numéro du symbole (p. 12).

$c_{j,k}$: symbole complexe représentant plusieurs bits à transmettre par une modulation OFDM. k est le numéro de la porteuse et j celui du symbole (p. 22).

$C(f)$: réponse fréquentielle du canal (p. 25).

$c(t)$: réponse impulsionnelle du canal (p. 20).

$\mathbf{d}$: unique symbole OFDM reçu après égalisation du canal (p. 67).

d_k : symbole reçu par un système monoporteuse (p. 17).

$d_{j,k}$: échantillons du symbole OFDM fréquentiel reçu numéro j, sur la porteuse k (p. 26).

E_b/N_0 : critère d'évaluation du rapport signal sur bruit, calculé en divisant l'énergie dépensée pour émettre un bit par la densité spectrale de puissance du bruit blanc gaussien (p. 33).

e_k : erreur en sortie du réseau pi-sigma pour l'exemple k (p. 62).

$\mathbf{e_{l_k}}$: vecteur contenant les erreurs des réseaux de la couche l d'un perceptron multicouches lorsque l'on présente l'exemple numéro k (p. 57).

$f()$: fonction d'activation d'un neurone de perceptron (p. 53).

f_0 : en monoporteuse : fréquence de la porteuse (p. 14). En multiporteuses : fréquence de la porteuse centrale (p. 22).

f_k : fréquence de la porteuse k, en modulation multiporteuses (p. 22).

$f_l()$: fonction d'activation des neurones de la couche d'un perceptron multicouches (p. 54).

$\mathbf{f}_l()$: extension vectorielle de $f_l()$ (p. 54).

$g_e(t)$: réponse impulsionnelle du filtre d'émission (p. 20).

$g_r(t)$: réponse impulsionnelle du filtre de réception (p. 17).

$h(t)$: forme d'onde utilisée pour interpoler le signal à moduler : rectangle de durée T_S (p. 13).

$h_{N_p}(t)$: forme d'onde utilisée pour réaliser une modulation OFDM : rectangle de durée T_S/N_p (p. 24).

$l(t)$: réponse impulsionnelle de l'égaliseur (p. 20).

N_a : nombre d'éléments dans la base d'apprentissage d'un système d'approximation de fonction (p. 45).

N_c : nombre de couches d'un perceptron multicouches (p. 54).

N_e : nombre d'entrées d'un système d'approximation de fonction, égal aux nombre de dimensions du vecteur d'entrée (p. 46).

N_g : nombre de neurones sur la première couche d'un réseau GRBF (p. 49).

n_l : nombre de neurones dans la couche l d'un perceptron multicouches (p. 54).

N_p : nombre de porteuses (p. 22).

N_s : nombre de sorties d'un système d'approximation de fonction, égal au nombre de dimensions du vecteur de sortie (p. 52).

$\mathbf{p_j}$: vecteur prototype du neurone j de la première couche d'un réseau GRBF (p. 49).

O_r : ordre d'un réseau d'ordre supérieur (HPU : p. 61, pi-sigma : p. 61, RPN :).

r_k : valeur de $R(f)$ à la fréquence f_k (p. 25).

$R(f)$: réponse fréquentielle de la chaîne de transmission complète (forme d'onde, filtre d'émission, canal, filtre de réception) (p. 25).

$r(t)$: réponse impulsionnelle de la chaîne de transmission (forme d'onde, filtre d'émission, éventuellement canal, filtre de réception) (p. 17).

$s(t)$: signal à l'entrée de l'amplificateur non linéaire (p. 66).

$s_0(t)$: signal à la sortie de l'amplificateur non linéaire (p. 66).

t_0 : instant d'échantillonnage du récepteur (p. 17).

T_S : durée d'un symbole à transmettre (p. 13).

$u_{j,k}$: coefficients de l'IDFT des $c_{j,k}$ en faisant varier k. Éléments du symbole OFDM temporel transmis (p. 24).

$u(t)$: signal modulé (monoporteuse p. 14, multiporteuse p. 22).

$u_c(t)$: partie réelle du signal à moduler en quadrature (p. 16).

$u_e(t)$: enveloppe complexe du signal à moduler en quadrature (p. 15).

$u_{ej}(t)$: enveloppe complexe du signal modulé correspondant au symbole OFDM numéro j (p. 23).

$u_j(t)$: signal modulé correspondant au symbole OFDM numéro j (p. 23).

$u_s(t)$: partie imaginaire du signal à moduler en quadrature (p. 16).

$v_{j,k}$: échantillons du symbole OFDM temporel reçu numéro j, k étant le numéro de l'échantillon, entre 0 et n_p. Ces échantillons sont fournis à la DFT dans le récepteur (p. 26).

$\mathbf{w_i}$: vecteur poids du neurone i de la première couche d'un réseau pi-sigma (p. 61).

$w_{i,j}$: poids liant le neurone j de la première couche et le neurone i de la seconde couche d'un réseau GRBF (p. 49).

$\mathbf{w_{i,j}}$: vecteur poids du neurone i de la première couche du réseau pi-sigma d'ordre j composant un réseau RPN (p. 63).

$\mathbf{W_l}$: matrice contenant tous les vecteurs poids des neurones de la couche l d'un perceptron multicouches (p. 54).

$\mathbf{w_{l,i}}$: vecteur poids du neurone i de la couche l d'un perceptron multicouches (p. 54).

$\mathbf{x_k}$: élément de l'exemple numéro k d'une base d'apprentissage pour un système d'approximation de fonction. Il correspond à l'entrée pour l'exemple $(\mathbf{x_k}, \mathbf{y_k})$ (p. 42).

$\mathbf{y_k}$: élément de l'exemple numéro k d'une base d'apprentissage pour un système d'approximation de fonction. Il correspond à la sortie pour l'exemple $(\mathbf{x_k}, \mathbf{y_k})$ (p. 42).

Bibliographie

[ANDR01] Aldo N. D'Andrea, Vincenzo Lottici, et Ruggero Reggiannini. «Nonlinear Predistortion of OFDM Signals over Frequency-Selective Fading Channels». *IEEE Transactions on Communications*. Vol. 49. N° 5. pp. 837-843. 2001.

[ATAR98] H. Atarashi et M. Nakagawa. «A Computational Cost Reduction Scheme for a Post-Distortion Type Nonlinear Distortion Compensator of OFDM Signals». *IEICE Transactions on Communications*. Vol. E81-B. N° 12. pp. 2334--2343. 1998.

[BALA95] P. Balay. *Application des techniques neuronales à la démodulation numérique. Thèse présentée devant l'Université de Rennes 1.* 1995.

[BAUM96] R. W. Bäuml, R. F. H. Fischer, et J. B. Huber. «Reducing the Peak-to-Average Power Ratio of Multicarrier Modulation by Selected Mapping». *IEE Electronics Letters*. Vol. 32. N° 22. pp. 2056-2057. 1996.

[BELI95] B. Beliczynski. «Incremental approximation by one-hidden-layer neural networks». *International Conference on Artificial Neural Networks*. Vol. 1. pp. 505-510. 1995.

[BELL02] M. Bellanger. *Traitement numérique du signal. Théorie et pratique.* ISBN: 2-10-006311-1. Dunod. 2002.

[BENE96] G. Di Benedetto et P. Mandarini. «An application of MMSE predistortion to OFDM systems». *IEEE Transactions on Communications*. Vol. 44. N° 11. pp. 1417-1420. 1996.

[BIGL84] E. Bigueri, A. Gersho, Richard D. Gitlin, et Tong Leong Lim. «Adaptive Cancellation of Nonlinear Intersymbol Interference for Voiceband Data Transmission». *IEEE Journal of Selected Areas in Communications*. Vol. 2. N° 5. pp. 765-777. 1984.

[BISH95] C. Bishop. *Neural Networks for Pattern Recognition*. ISBN: 0198538642. Oxford University Press. 1995.

[BOOR78] C. de Boor. *A Practical guide to splines*. ISBN: 0-387-90356-9. Springer-Verlag, New York. 1978.

[BOSM96] Sander Bosman. *Locally weighted approximations: yet another type of neural network. Master Thesis, University of Amsterdam.* 1996.

[BOUC99] Steven Bouchired, Daniel Roviras, et Francis Castanié. «Eqalisation of Satellite Mobile Channels with Neural Network Techniques». *Space Communications*. Vol. 15. N° 4. pp. 209-220. 1999.

[BURE91] G. Burel. *Réseaux de neurones en traitement d'images: des modèles théoriques aux applications industrielles. Thèse présentée devant l'Université de Bretagne Occidentale.* 1991.

[CADE02] *SPW Communications Library - Floating Point Reference. Product Version 4.8.* Cadence Design Systems, Inc.. 2002.

[CHAN98] Shun-Hsyung Chang et Shou-Yi Lu. «Neural-net Decoders for Linear Block Codes». *Journal of the Chinese Institute of Electrical Engineering*. Vol. 5. N° 4. pp. 333-343. 1998.

[CHAN98] P Chandra Kumar, P Saratchandran, et N Sundararajan. «Minimal Radial Basis Function Neural Networks for Non-linear Channel Equalisation». *IEE Proceedings of Image Signal Processing*. Vol. 147. N° 5. pp. 428-434. 2000.

[COHE92] Gérard Cohen, Jean-Louis Dornstetter, et Philippe Godlewski. *Codes correcteurs d'erreurs. Une introduction au codage algébrique.* ISBN: 2-225-82538. Masson. 1992.

[COST99] E. Costa, M. Midrio, et S. Pupolin. «Impact of Amplifier Nonlinearities on OFDM Transmission System Performance». *IEEE Communications Letters*. Vol. 3. N° 2. pp. 37-39. 1999.

[DAVI97] J.A. Davis et J. Jedwab. «Peak-to-mean power control and error correction for OFDM transmission using Golay sequences and Reed-Muller codes». *IEE Electronics Letters*. Vol. 33. N° 4. pp. 267-268. 1997.

[ERDO01] D. Erdogmus, D. Rende, J.C. Principe, et T.F. Wong. «Nonlinear Channel Equalization Using Multilayer Perceptrons with Information-Theoretic Criterion». *Proceedings of 2001 IEEE Signal Processing Society Workshop*. pp. 443-451. 2001.

[FLAK98] G. W. Flake. «Square unit augmented, radially extended, multilayer perceptrons». *Neural Networks: Tricks of the Trade, LNCS 1524*. ISBN: 3-540-65311-2. pp. 145-163. 1998.

[FLOC95] B. Le Floch, M. Alard, et C. Berrou. «Coded Orthogonal Frequency Division Multiplex». *Proceedings of the IEEE*. Vol. 83. N° 6. pp. 982-996. 1995.

[FOUC02] Christophe Foucher. *Analyse et amélioration d'algorithmes neuronaux et non neuronaux de quantification vectorielle pour la compression d'images. Thèse présentée devant l'Université de Rennes 1.* 2002.

[FRIE81] J.H. Friedman et W. Stuetzle. «Projection pursuit regression». *Journal of the American Statistical Association*. N° 76(376). pp. 817-823. 1981.

[FRIT94] B. Fritzke. «Fast learning with incremental RBF networks». *Neural Processing Letters*. Vol. 1. N° 1. pp. 2-5. 1994.

[GILE94] C.L. Giles et T. Taxwell. «Learning, Invariance and Generalization in High-Order Neural Networks». *Selected Papers on Optical Neural Networks*. ISBN: 0-8194-1578-2. pp. 344-350. 1994.

[GLAV96] Alain Glavieux et Michel Joindot. *Communications numériques. Introduction.* ISBN: 2-225-85194. Masson. 1996.

[GROS93] R. Gross et D. Veeneman. «Clipping distortion in DMT ADSL systems». *Electronic Letters*. Vol. 29. N° 24. pp. 2080-2081. 1993.

[GUO02] Y. Guo et J. Cavallaro. «On the Reduction of Peak-to-Average Power Ratio in Adaptively Companded OFDM Signals with Non-Linear Power Amplifier». *3G Wireless, World Wireless Congress*. 2002.

[HENK00] W. Henkel. «Analog Codes for Peak-to-Average Ratio Reduction». *3rd ITG Conference Source and Channel Coding*. 2000.

[HENK00b] Werner Henkel et Björn Wagner. «Another Application for Trellis Shaping: PAR Reduction for DMT (OFDM)». *IEEE Transactions on Communications*. Vol. 48. N° 9. pp. 1471-1476. 2000.

[HORN93] K. Hornik. «Some New Results on Neural Network Approximation». *Neural Networks*. Vol. 6. N° 8. pp. 1069-1072. 1993.

[JEDW97] J. Jedwab. «Comment:"M-Sequences for OFDM Peak-to-Average Power Ratio Reduction and Error Correction"». *IEE Electronics Letters*. Vol. 33. N° 15. pp. 1293-1294. 1997.

[JIAN95] Jian-ping Yu, Yu-biao Zhao, et Xin-mei Wang. «On Neural Decoding for Some Cyclic Codes». *IEEE International Symposium on Information Theory*. pp. 417. 1995.

[JOYD92] Joydeep Ghosh et Yoan Shin. «Efficient Higher-order Neural Networks for Classification and Function Approximation». *Int. J. Neural Systems*. Vol. 3. N° 4. pp. 323-350. 1992.

[KURK95] Věra Kůrková et Bartłomiej Beliczyński. «An Incremental Learning Algorithm for Gaussian Radial-Basis-Function Approximation». *Second International Symposium on Methods and Models in Automation and Robotics*. pp. 675-680. 1995.

[LANG00] Fabien Langlet-Cauët, Daniel Roviras, Francis Castanié, et Jacques Sombrin. «Implantation des Réseaux de Neurones: Architecture et Application aux Communications par Satellite». *AAA2000 (Adéquation Algorithme Architecture)*. pp. 93-98. 2000.

[LI97] X. Li et J. A. Ritcey. «M-sequences for OFDM peak-to-average power ratio reduction and error correction». *IEE Electronics Letters*. Vol. 33. N° 7. pp. 554-555. 1997.

[LOUE00] Yves Louët. *Mise en œuvre et performances des codes de Reed-Muller pour la réduction du facteur de crête dans la modulation O.F.D.M.. Thèse présentée devant l'Université de Rennes 1*. 2000.

[LUND97] T. Lundin et P. Moerland. «Quantization and Pruning of Multilayer Perceptrons: Towards Compact Neural Networks». *IDIAP*. N° IDIAP-Com 97-02. 1997.

[MAN94] Marcia Man Peng. *Neural Networks Applications in Linear and Nonlinear Channel Equalization. PhD Thesis, Northeastern University, Boston, Massachusetts.* 1994.

[MARQ63] D. W. Marquardt. «An algorithm for least-squares estimation of nonlinear parameters». *Journal of the Society for Industrial and Applied Mathematics*. Vol. 11. N° 2. pp. 431-441. 1963.

[MERC98] Sergio Merchá González, Ana García Armada, et Jose Luís García García. «OFDM performance in amplifier nonlinearity». *IEEE Transactions on Broadcasting*. Vol. 44. N° 1. pp. 106-114. 1998.

[MOZA98] N. Mozayyani, A. R. Baig, et G. Vaucher. «A fully-neural solution for on-line handwritten character recognition». *IEEE Int. Joint Conf. on Neural Networks*. Mai 1998.

[MULL97] S. H. Müller et J. B. Huber. «A Novel Peak Power Reduction Scheme for OFDM». *PIMRC'97*. Vol. 3. pp. 1090-1094. 1997.

[NISH96] H. Nishijima, M. Okada, et S. Komaki. «A sub-optimum non-linear distortion compensation scheme for orthogonal multi-carrier modulation systems». *Proceedings of PIMRC'96*. pp. 45-48. 1996.

[PARK91] J. Parks et I.W. Sandberg. «Universal Approximation using Radial-Basis Function Network». *Neural Computation*. Vol. 3. N° 2. pp. 246-257. 1991.

[PAUL96] Mathias Pauli et Hans-Peter Kuchenbecker. «Minimization of the Intermodulation Distortion of a Nonlinearly Amplified OFDM Signal». *Wireless Personal Communications*. N° 4. pp. 93-101. 1996.

[POGG89] T. Poggio et F. Girosi. «A Theory of Networks for Approximation and Learning». *A. I. Memo*. N° 1140. 1989.

[RAPP91] C. Rapp. «Effects of HPA-nonlinearity on 4-DPSK-OFDM-signal for a digital sound broadcasting system». *Proceedings of 2nd European Conference on Satellite Communications*. pp. 179-184. 1991.

[RUME86] D. E. Rumelhart et J.L. Mc Clelland. *Parallel Distributed Processing*. *Explorations in the Microstructure of Cognition.* ISBN: 0-262-68053-X. 1986.

[SALE87] A. Saleh et R. Valenzuela. «A Statistical Model for Indoor Multipath Propagation». *IEEE Journal on Selected Areas in Communications*. Vol. 5. N° 2. pp. 128-137. 1987.

[SHI96] Qun Shi. «OFDM in bandpass nonlinearity». *IEEE Transactions on Consumer Electronics*. Vol. 42. N° 3. pp. 253-258. 1996.

[SHIN91] Y. Shin et J. Ghosh. «The Pi-sigma Network : An Efficient Higher-Order Neural Network for Pattern Classification and Function Approximation». *Proceedings IJCNN*. pp. 13-18. 1991.

[SHIN92] Y. Shin et J. Ghosh. «Approximation of Multivariate Functions Using Ridge Polynomial Networks». *Proceedings IJCNN*. Vol. 2. pp. 380-385. 1992.

[SHIN95] Yoan Shin et Joydeep Ghosh. «Ridge Polynomial Networks». *IEEE Transactions on neural networks*. Vol. 6. N° 2. pp. 610-622. 1995.

[TANG91] P.T.P. Tang. «Table lookup algorithms for elementary functions and their error analysis». *Proceedings of the 10th IEEE Symposium on Computer Arithmetic*. pp. 232-236. 1991.

[TELL98] José Tellado et John M. Cioffi. «Peak Power Reduction for Multicarrier Transmission». *Proceedings of Globecom 98*. 1998.

[TERT02] S. Tertois, A. Le Glaunec, et G. Vaucher. «Compensating the non linear distortions of an OFDM signal using neural networks». *Recent Trends in Multimedia Information Processing. Proceedings of IWSSIP'02.* ISBN: 981-238-243-7. pp. 484-488. 2002.

[TERT03] S. Tertois, Y. Louët, G. Vaucher, et A. Le Glaunec. «Réduction des effets des non-linéarités du signal OFDM à l'aide d'un réseau de neurones d'ordre supérieur». *Actes de Setit 2003*. ISBN: 9973-41-685-6. pp. 36. 2003.

[TERT03b] S. Tertois et Y. Louët. «Non linear compensation of an OFDM signal in the time domain with a neural network». *ISSPIT'03. article soumis.*

[VAPN98] V. Vapnik, S. Golowich, et A. Smola. «Support vector method for function approximation, regression estimation, and signal processing». *Advances in Neural Information Processing Systems*. Vol. 9. pp. 281-287. 1996.

[WILK95] T. A. Wilkinson et A. E. Jones. «Minimisation of the Peak to Mean Envelope Power Ratio of Multicarrier Transmission Schemes by Block Coding». *IEEE VTC'95*. N° 2. pp. 825-829. 1995.

[YAMA98] S. Yamasaki et H. Tanaka. «A Nonlinear Distortion Compensation on Layered Multicarrier Modulation Using Iterative Restoration». *IEICE Transactions Fundamentals*. Vol. E81-A. N° 10. pp. 2109-2117. 1998.

Glossaire

ADSL (Asymetric Digital Subscriber Line) : Système de communication haut débit asymétrique sur liaison filaire téléphonique, employée principalement pour les connections de particuliers et d'entreprises à Internet.

AM/AM (Amplitude Modulation/Amplitude Modulation) : Non linéarité d'amplitude (p. 66)

AM/PM (Amplitude Modulation/Phase Modulation) : Non linéarité de phase (p. 66)

ARQ (Automatic Repeat reQuest) : Protocole de gestion d'erreurs de transmission. Les erreurs sont détectées par le code correcteur. Lorsqu'une erreur est détectée, le message transmis est répété (p. 18)

BBAG (Bruit Blanc Additif Gaussien) : Modèle de canal dans lequel le bruit est modélisé par une variable aléatoire gaussienne et ajouté au signal transmis (p. 33)

CF (Crest Factor) : Facteur de crête. Racine carrée du PMEPR (p. 28)

COFDM (Coded Orthogonal Frequency Division Multiplexing) : Utilisation conjointe d'une modulation OFDM et d'un code correcteur d'erreurs (p. 27)

DFE (Decision Feedback Equilizer) : Égaliseur de canal linéaire travaillant dans le domaine temporel (p. 100)

DFT (Dirscrete Fourier Transform) : Transformée de Fourier discrète (p. 26)

DSP (Densité Spectrale de Puissance) : Courbe représentant le spectre d'un signal (p. 14)

FEC (Forward Error Correction) : Protocole de gestion d'erreurs de transmission. Les erreurs sont corrigées par le code correcteur (p. 18)

GRBF (Gaussian Radial Basis Functions) : Base de fonctions radiales gaussiennes. Famille de réseaux de neurones qui se basent sur la distance entre le vecteur d'entrée et un ensemble de prototypes (p. 48)

HPU (Higher-order Processing Unit) : Couche de perceptron qui calcule sa sortie à l'aide d'un polynôme, au lieu d'un simple produit scalaire. Permet de réaliser des réseaux d'ordre supérieur (p. 60)

IBO (Input Back Off) : Recul d'un amplificateur. Rapport entre la puissance de saturation ramenée à l'entrée et la puissance moyenne du signal en entrée (p. 30)

IDFT (Inverse Discrete Fourier Transform) : Transformée de Fourier discrète inverse (p. 24)

IES (Interférence Entre Symboles) : Perturbation introduite par le canal ou les filtres de transmission qui aboutit à un mélange entre symboles successifs (p. 17)

MDP (Modulation De Phase) : Codage binaire à symbole complexe dans lequel l'information est codée dans la phase du symbole (p. 12)

MAQ (Modulation d'Amplitude en Quadrature) : Codage binaire à symbole complexe dans lequel l'information est codé à la fois dans la partie réelle et la partie imaginaire du symbole (p. 13)

OFDM (Orthogonal Frequency Division Multiplexing) : Modulation multiporteuses utilisant une base orthogonale de sous-porteuses (p. 22)

PMC (Perceptron MultiCouche) : Famille de réseaux de neurones, dont chaque neurone effectue un produit scalaire entre son vecteur d'entrée et son vecteur poids (p. 53)

PMEPR (Peak to Mean Envelope Power Ratio) : Rapport entre la puissance maximale et la puissance moyenne d'un signal temporel (p. 28)

PTS (Partial Transmit Sequences) : Méthode de réduction du facteur de crête qui consiste à séparer le symbole OFDM en plusieurs blocs et à appliquer une rotation différente sur chaque bloc (p. 35)

RPN (Ridge Polynomial Network) : Généralisation du réseau Pi-Sigma, approximateur universel (p. 63)

RSB (Rapport Signal sur Bruit) : Voir SNR.

SNR (Signal to Noise Ratio) : Rapport signal sur bruit. Puissance du signal utile divisé par la puissance du bruit (p. 73)

SPW (Signal Processing Workshop) : Outil écrit par l'entreprise Cadence permettant de simuler efficacement des systèmes de communication numérique (p. 72)

SQUARE-MLP (SQuare Unit Augmented Radially Extended MultiLayer Perceptron) : PMC auquel on fournit en plus des entrées le carré de ces dernières, afin d'améliorer ses capacités d'approximation (p. 59)

SSPA (Solid State Power Amplifier) : Modèle non linéaire d'amplificateur à semi-conducteurs (p. 29, p. 66 et équation (3.2))

TEB (Taux d'Erreur Binaire) : Évaluation de la qualité d'une transmission numérique. Nombre de bits d'information erronés divisé par le nombre de bits transmis (p. 33)

TOP (Tube à Ondes Progressives) : Voir TWT.

TWT (Travelling Wave Tube) : Tube à ondes progressive. Modèle d'amplificateur non linéaire (p. 66 et équation (3.2))

Index

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

Z