REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE YAHIA FARES DE MEDEA

Faculté de **T**echnologie

Département du **G**énie **E**lectrique

Mémoire de fin d'études de Master

Filière : Télécommunication

Spécialité : Systèmes de Télécommunication

Suppression de Bruit par Filtrage Adaptatif Utilisant de Réseaux de Neurones

Présentés par :

Ghernaout Soumia

Salem Ahlem

Proposés et Dirigés par :

Docteur Laidi Kamel

Promotion 2022

Remerciements

Au terme de ce mémoire, nous tenons remercier d'abord et avant tout ALLAH qui nous a donné la capacité, la force, le courage et la patience d'accomplir ce travail.

Nous tenons également à exprimer mes sincères remerciements au notre promoteur M.Laidi Kamal, pour leur encadrement, surtout pour leur confiance et leurs précieux conseils qui nous ont beaucoup aidés.

Enfin, nous tenons à exprimer ma profonde gratitude à nos familles et nos amis qui nous ont toujours soutenues. Et à tous ceux qui ont contribué de près ou de loin à la réalisation de ce modeste travail.

Dédicaces

A tout étudiant en sciences qui cherche à acquérir des connaissances et à apporter son équilibre scientifique et culturel

Aux deux plus grandes personnes qui ont illuminé mon chemin et ont été mon soutien après ALLAH. A ceux qui m'ont appris que le monde est un combat et que son arme est la science et la connaissance, mes chers parents, que Dieu les protèges

A toute la généreuse famille qui m'a soutenu et qui sont toujours, mes chers frères et chères sœurs,

A ma copine Ahlem et à tous mes amis qui ont partagé mes moments

À tous mes collègues du domaine du génie des procédés pharmaceutiques à tous les cadres du Département de génie éclectique et à tous les travailleurs de la Faculté de technologie de l'Université Dr. Yahya Fares à Medea

A tous ceux qui ont eu un impact dans ma vie et à tous ceux que mon cœur a aimés et qui ont été oubliés par ma plume

Je dédie ce modeste travail

Ghernaou Soumia



Dédicaces

Je dédie ce travail a ma mère pour son amour , ses encouragements et ses sacrifiées A mo père pour son soutien son affection et la confiance qu'il m'a accordé a la mémoire de ma chère grande mère à tous les membres de ma famille Atous mes amis et tous ceux qui m'aiment

Salem ahlem

ملخص

العديد من الإشارات يشوبها تداخل أو ضوضاء غير مرغوب فيها. لذلك، من الضروري التخلص من هذه المكونات المزعجة. يكون الترشيح التقليدي فعالاً إذا تم فصل أطياف الإشارة والضوضاء. إذا لم يكن الأمر كذلك ، فهناك طريقة أخرى للتدخل كحل ممتاز وهي التصفية التكيفية التي تضبط تلقائيًا استجابة الاندفاع.

في هذا المشروع ، قمنا بتنفيذ نظام إلغاء الضوضاء التكيفي باستخدام الشبكات العصبية كخوارزميات التكيف.

اولا تمت دراسة متعمقة للترشيح التكيفي والشبكات العصبية. ثم تم تنفيذ تطبيق لقمع الضوضاء.

تم إجراء العديد من الاختبارات لإشارات مختلفة بتردد ثابت، ثم على تردد متغير. النتائج التي تم الحصول عليها مرضية وقابلة للمقارنة مع تلك الخاصة بخوارزميات الكلاسيكية.

الكلمات المفتاحية : الترشيح التكيفي – الشبكات العصبية- قمع الضوضاء.

Résumé

De nombreux signaux temporels sont entachés d'interférences ou de bruits indésirables. Par conséquent, il est nécessaire d'éliminer ces composants gênants. Le filtrage conventionnel est efficace si les spectres du signal et du bruit sont séparés. Si ce n'est pas le cas, un autre moyen d'intervenir comme une excellente solution est le filtrage adaptatif qui ajuste automatiquement sa réponse impulsionnelle.

Dans ce projet, nous avons mis en œuvre un système de suppression de bruit adaptatif utilisant des réseaux de neurones comme algorithmes d'adaptation.

Une étude approfondie du filtrage adaptatif et des réseaux de neurones a été étudiée. Ensuite une application à la suppression de bruit a été effectuée.

Plusieurs tests ont été effectués pour différents signaux à fréquence fixe, puis à fréquence variable. Les résultats obtenus sont satisfaisants et comparables avec ceux des Algorithmes classiques RLS, LMS et NLMS.

Mots clé : filtrage adaptatif – réseaux de neurones – suppression de bruit.

Abstract

Many signals are marred by unwanted interference or noise. Therefore, it is necessary to eliminate these troublesome components. Conventional filtering is effective if the signal and

noise spectrums are separated. If this is not the case, another way to intervene as an excellent solution is adaptive filtering which automatically adjusts its impulse response.

In this project, we implemented an adaptive noise cancellation system using neural networks as adaptation algorithms.

A depth survey of adaptive filtering and neural networks has been studied. Then an application to noise suppression was carried out.

Several tests were carried out for different signals at fixed frequency, then at variable frequency. The obtained results are satisfactory and comparable with those of the classic RLS, LMS and NLMS algorithms.

Keys words: adaptive filtering – neural networks – a noise suppression.

Sommaire:

Remerciment	
Dedicace	
Résumé	
Sommaire	
Liste des Figure	
Liste des tableaux	
Introduction Général	1
Chapitre I : Les filtres adaptatif	
I.1. Introduction	3
I.2. Filtrage numérique	3
I.3.Filtrage adaptatif	3
I.4. Principe d'un filtrage adaptatif	3
I.5. Rôle du filtre adaptatif	5
I.6. Applications du filtrage adaptatif	5
I.6.1. Identification de systèmes	6
I.6.2. Prédiction	6
I.6.3. Modélisation inverse	7
I.6.4. Annulation d'interférences	7
I.7. Du filtrage de Wiener au filtrage adaptatif	8
I.7.1. Filtrage de Wiener	8
I.7.2. Principe d'orthogonalité	9
I.7.3. Equation de Wiener – Hopf	10
I.8. Définition d'un algorithme	11

I.9. Le choix de l'algorithme	11
I.10. Algorithme du moindre carré moyen LMS	11
I.11. Dérivation et l'algorithme LMS	12
I.12. Algorithme de moindres carrés normalisés (NLMS)	13
I.13. Dérivation de l'algorithme NLMS	13
I.14.Algorithmes NLMS proportionnel	15
I.14.1. Algorithme PNLMS	16
I.14.2. Algorithme PNLMS amélioré (IPNLMS)	17
I.15. L'algorithme RLS	17
I.15.1. Dérivation de l'algorithme RLS	18
I.15.2.Mise en œuvre de l'algorithme RLS	19
I.16.Critères de comparaison entre les algorithmes adaptatifs	19
I.17. Conclusion	19
Chapitre II : Les réseaux de neurones artificiel	
II.1 Introduction	21
II.2 Réseaux de neurones	21
II.2.1 Neurone biologique	21
II.2.1.1 Structure de neurone	22
II.2.2 Neurone formel	22
II.2.2.1 Poids	23
II.2.2.2 Seuil	23
II.2.2.3 Fonction d'activation	23
II .3 Réseau de neurones artificiels	24
II.3.1 Définition	24
II.3.2 Fonctionnement de RNAs	25
II .3.3 Historique de RNAs	25
II.3.4 Perceptron multicouche	26
II.3.4.1 Algorithme de rétro-propagation	27

II.3.4.1. a. Algorithme de gradient	28
II.3.4.1.b. Étapes de rétro-propagation	28
II.3.5 Réseaux de neurones à fonction de base radiale (RBF)	30
II.4 Théorème d'approximation universelle	31
II.5 Avantages et inconvénients des réseaux de neurones	31
II.5.1 Avantages	31
II.5.2 Inconvénients	31
II.6 Conclusion	32
Chapitre III : Filtres adaptatif utilisant les réseaux de neurones	
III.1. Introduction	33
III.2 Les réseaux de neurones utilisés comme filtres non lineaires	34
III.2.1 Modèle de neurone formel	34
III .2.2 Réseaux de neurones non bouclés utilisés comme filtre transverse	35
III.2.3 Réseaux de neurones bouclés utilisés comme filtres récursifs	37
III.3 Apprentissage des réseaux de neurones adaptatif	39
III. 3.1 Critère d'optimisation	39
III.3.2 Algorithmes adaptatifs fondés sur l'estimation du gradient de la fonction de coût	41
III.3.3 Evaluation du gradient dans un réseau non bouclé	44
III.3.3.1 Calcul direct	44
III.3.3.2 Calcul par rétro propagation	45
III .4 Equivalence des notions de récurrence temporelle et de dépliement spatial	46
III.5 Algorithmes pour l'adaptation des réseaux bouclés	46
III.5.1 Dépliement du réseau bouclé en un nombre fini de copies	46
III.5.2 Une nouvelle famille d'algorithmes adaptatifs	47
III.5.3 Classification des algorithmes existants en filtrage adaptatif et pour l'apprentissage des réseaux deneurones.	49

III . 6 Concusion	50
Chapitre IV : Application à la suppression de bruit	
IV.1 Introduction	51
IV.2Généralité sur Filtres de réseau de neurones adaptatifs	51
IV.2.1 Foncons adaptatives	51
IV.2.2 Modèle de neurone linéaire	51
IV.2.3 Architecture de réseau linéaire adaptatif	52
IV.2.4 Erreur des moindres carrés moyens	57
IV.2.5 Algorithme LMS (learnwh)	57
IV.2.6 Filtrage adaptatif (adapter)	58
IV .3 la prédiction	64
IV.4 Suppression d'une interférence	65
IV.4.1 Domaines d'application de l'annulation de bruit	66
IV.4.2 Méthodes de l'annulation de bruit	66
IV.4.3 Principe de la méthode ANC	66
IV.4.4 Explication de la méthode.	67
IV.5 Résultats de simulation	69
IV.6. Discussion	73
IV .7. Conclusion	73

Liste des figures

Figure I.1 *Principe d'un filtre adaptatif.*	**4**
Figure I.2 *Structure complète d'un système d'annulation d'écho.*	**5**
Figure I.3.*Identification du système.*	**6**
Figure I.4 *Filtre prédictif direct d'erreur.*	**6**
Figure I.5 *Modélisation inverse.*	**7**
Figure I.6 *Annulation d'interférences*	**8**

Figure I.7 *Schéma du filtrage de Wiener.*	**8**
Figure II.1 *Schéma d'un neurone biologique*	**22**
Figure II.2 *Neurone formel*	**23**
Figure II .3 *Perceptron à deux couches.*	**26**
Figure II.4 *perceptron multicouche.*	**27**
Figure II .5 *Mécanisme d'apprentissage.*	**28**
Figure II.6 *réseaux à fonction de base radiale.*	**31**
Figure III. 3 *Neurone formel*	**34**
Figure III.4 *Réseau de neurones non bouclé complètement connecté utilisé comme filtre transverse non linéaire*	**36**
Figure III. 3 *Réseau de neurones bouclé (modèle NARMAX) utilisé comme filtre récursif non linéaire.*	**38**
Figure III.4 *Surfaces d'erreur I(n).*	**42**
Figure IV.1 *Modèle de neurone linéaire*	**51**
Figure IV.2 *Fonction de transfert linéaire*	**52**
Figure IV.3 *Réseaux linéaire adaptatif*	**53**
Figure IV.4 *Couche de neurones*	**54**
Figure IV.5 *ADALINE unique*	**55**
Figure IV.6 *Filtrage adaptatif*	**58**
Figure IV.7 *une ligne à retard à prises avec un réseau ADALINE*	**59**
Figure IV.8 *réseau linéaire en utilisant linearlayer.*	**60**
Figure IV.9 *Filtre prédictif*	**63**
Figure IV. 10 : *Principe de la méthode « ANC »*	**67**
Figure IV.11 : *Montage de filtre adaptatif pour l'annulation de bruit.*	**67**
Figure IV .12 *Réduction du bruit dans un signal sinus à fréquence variable*	**69**
Figure IV .13 *Réduction du bruit dans un signal a dents de scie (sawtooth)*	**70**
Figure IV.14 *Réduction du bruit dans un signal squar*	**71**
Figure IV.15 *Réduction du bruit dans un signal(le signal variable*signal square)*	**72**

Liste des Tableaux

Tableau II.1 : *Fonctions d'activation.*	**24**
Tableau III.1 : *classification des algorithmes à partir des initialisations du système d'argument* $m \in [n-N_c+2, n]$ *pour la modification des coefficients à l'instant n : calcul direct. Pour les lignes 1, 2 et 3, on utilise les valeurs calculées à l'instant n – 1 pour initialiser la première copie du système d'argument* $m = n - N_c + 1$.	**47**
Tableau III. 2 *: classification des algorithmes à partir des initialisations du système d'argument* $m \in [n-N_c+2, n]$ *pour la modification des coefficients à l'instant n : rétro propagation. Pour la ligne 1, on utilise les valeurscalculées à l'instant n – 1 pour initialiser la première copie du système d'argument* $m = n - N_c + 1$.	**48**

Introduction générale

Introduction générale

Le traitement du signal est la discipline qui développe et étudie les techniques de traitement (filtrage, amplification...), d'analyse et d'interprétation des signaux. Elle fait donc largement appel aux résultats de la théorie de l'information, des statistiques ainsi qu'à de nombreux autres domaines des mathématiques appliquées**.**

On appelle filtre numérique, un système utilisé pour modifier la distribution fréquentielle d'un signal numérique d'entrée en le transformant en un signal numérique désiré en sortie. Les filtres numériques sont caractérisés par leur précision, fiabilité, stabilité, adaptabilité et facilité de commande. Le filtre numérique peut être de type filtre à réponse impulsionnelle finie (FIR : Finite Impulse Response) ou filtre à réponse impulsionnelle infinie (IIR: Infinite Impulse Response).

Un filtre adaptatif est un filtre numérique dont les coefficients se modifient eux-mêmes en fonction des signaux extérieurs. Il est utilisé chaque fois qu'un environnement est mal connu ou changeant, ou pour supprimer des perturbations situées dans le domaine des fréquences du signal utile .

L'un des défis de l'homme consiste à reproduire son propre mode de raisonnement et de comportement. De ce désir a généré des réseaux de neurones, ils constituent une famille de fonctions non linéaires paramétrées, Les réseaux de neurones sont définis comme des approximateurs adaptables capables de reproduire le comportement complexe des systèmes non linéaires.

Dans la mesure où l'approximation de fonction et l'identification de signaux ou de systèmes sont au cœur des préoccupations du filtrage adaptatif, il est naturel de cherche à évaluer l'apport potentiel des réseaux de neurones dans ce domaine, notamment pour la réalisation d'opération de filtrage non linéaire en traitement du signal.

Dans ce contexte et afin d'exploiter la grande puissance des réseaux de neurones dans les problèmes d'adaptation (apprentissage). Notre travail consiste à exploiter cette puissance pour ajuster les coefficients du filtre adaptatif afin de l'appliquer à la suppression du bruit, donc l'extraction d'un signal utile noyé dans un bruit.

Afin d'atteindre cet objectif, nous avons organisé notre mémoire comme suit :

Dans le premier chapitre nous présentons un rappel théorique sur le filtrage numérique, en particulier le filtrage adaptatif (principe, application et rôle d'un filtre adaptatif), puis le filtrage de Wiener, définition d'un algorithme et le choix d'un algorithme adaptatif (LMS, NLMS,

PNLMS, RLS) et à la fin quelques évaluations de quelques critères de comparaison entre les algorithmes adaptatifs.

Dans le deuxième chapitre nous présentons un rappel théorique sur Réseaux de neurones (Neurone biologique, Neurone formel) et réseau de neurones artificiels (RNAs) (définition, fonctionnement de RNAs, historique de RNAs, perceptron multicouche, avantages et inconvénients des réseaux de neurones.

Dans le troisième chapitre, nous abordons les réseaux de neurones non bouclés utilisés comme filtres transverses et apprentissage des réseaux de neurones adaptatifs, équivalence des notions de récurrence temporelle et de dépliement spatial et algorithmes pour l'adaptation des réseaux bouclés.

Le quatrième chapitre explicite l'application du filtrage adaptatif basé sur les réseaux de neurones comme outil d'adaptation à la suppression du bruit d'un signal utile.

Enfin une conclusion générale qui résume les conclusions tirées de ce travail et l'efficacité d'utiliser les réseaux de neurones comme moyen d'ajustement des coefficients d'un filtre adaptatif.

Chapitre I: Les filtres adaptatifs

I.1. Introduction

De façon générale les filtres adaptatifs sont des systèmes appliqués sur des données bruitées pour obtenir une information utile à un certain instant t.

Les méthodes adaptatives ont connu un essor considérable depuis les années 60, dû au développement du traitement numérique et à l'augmentation constante de la puissance des processeurs de traitement, permettant la mise en œuvre en temps réel d'algorithmes de plus en plus sophistiqués, à des cadences de plus en plus rapides. Elles sont arrivées à une certaine maturité aussibien en ce qui concerne le développement et l'implémentation des algorithmes, que du point de vue des outils théoriques d'étude des performances. [1]

Dans ce chapitre nous présentons un rappel théorique sur le filtrage numérique puis le filtrage adaptatif (le principe, l'application et le rôle d'un filtre adaptatif), puis le filtrage de Wiener, et définition d'un algorithme, le choix de l'algorithme et l'algorithme adaptatif (LMS, NLMS, PNLMS, RLS) et à la fin quelques évaluations de quelques critères de comparaison entre les algorithmes adaptatifs.

I.2. Filtrage numérique

On appelle filtre numérique, un système utilisé pour modifier la distribution fréquentielle d'un signal numérique d'entrée en le transformant en un signal numérique désiré en sortie. Les filtres numériques sont caractérisés par leur précision, fiabilité, stabilité, adaptabilité et facilité de commande. Le filtre numérique peut être de type filtre à réponse impulsionnelle finie (FIR : Finite Impulse Response) ou filtre à réponse impulsionnelle infinie (IIR : Infinite Impulse Response). En principe, il peut utiliser n'importe quelle structure, directe, en cascade, en treillis, en échelle, etc. [2].

I.3.Filtrage adaptatif

Un filtre adaptatif est un filtre numérique dont les coefficients se modifient euxmêmes en fonction des signaux extérieurs. Il est utilisé chaque fois qu'un environnement est mal connu ou changeant, ou pour supprimer des perturbations situées dans le domaine des fréquences du signal utile [2]

I.4. Principe d'un filtrage adaptatif

Le principe de filtrage adaptatif appliqué au filtrage d'un système, repose classiquement sur l'hypothèse que le système est modélisé exactement par un filtre à Réponse Impulsionnelle Finie (FIR : Finie Impulse Response) de L coefficients. Le filtre adaptatif, modélisé par un filtre FIR de taille idéalement égale à celle du système inconnu, a pour rôle de fournir une estimée du signal inconnu, l'estimée étant obtenue en réalisant une opération de filtrage du signal par le système.

La technique de filtrage adaptatif se décompose classiquement en deux 2 étapes :

1. une étape de filtrage qui permet d'obtenir une estimée du signal inconnu en convoluant le signal d'entrée avec les coefficients du filtre adaptatif. L'erreur d'estimation est ensuite utilisée dans la partie adaptation pour mettre à jour les coefficients de filtre.
2. Une étape d'adaptation qui permet d'ajuster les coefficients du filtre adaptatif suivant un algorithme donné.

   L'algorithme de filtrage adaptatif permet de calculer les coefficients du filtre w de façon à ce que la différence entre le signal $\{d\}$ et l'actuelle sortie du filtre $\{y\}$ soit minimisée au sens d'un critère statistique préalablement défini.
   De manière générale, l'algorithme d'adaptation se présente sous la forme vectorielle suivante : **[2]**

$$\begin{pmatrix}\text{Vecteur de}\\ \text{nouveaux}\\ \text{coefficients}\\ \text{du filtre}\end{pmatrix} = \begin{pmatrix}\text{vecteur des}\\ \text{anciens}\\ \text{coefficients}\\ \text{du filtre}\end{pmatrix} + (\text{pas d'adaptation}) \cdot \begin{pmatrix}\text{échantillons}\\ \text{du signal}\\ \text{d'erreur}\end{pmatrix} \cdot \begin{pmatrix}\text{vecteur}\\ \text{du signal}\\ \text{d'entrée}\end{pmatrix}$$

Les différents signaux qui sont utilisés dans le filtre adaptatif qui estreprésenté à la figure (I.1) sont

x (n) : Est le signal de l'entrée du filtre.

w (n) : Sont les coefficients du filtre adaptatif.

y(n) :Est le signal de la sortie du filtre adaptatif.

d(n) : Est le signal de référence et qui est la sortie du filtre inconnu H.

e (n) : Est le signal d'erreur (c'est la différence entre d(n) et y(n)).

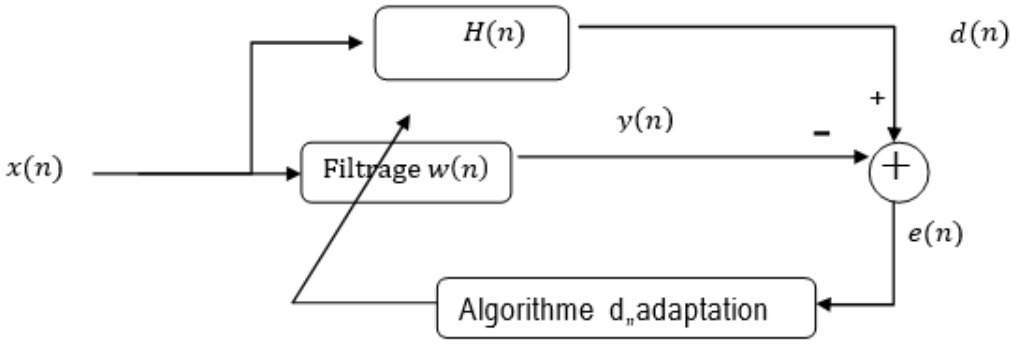


Figure I.1 *Principe d'un filtre adaptatif.*

Le cas de l'application annulation d'écho, ces signaux sont représentés dans la figure (I-1)

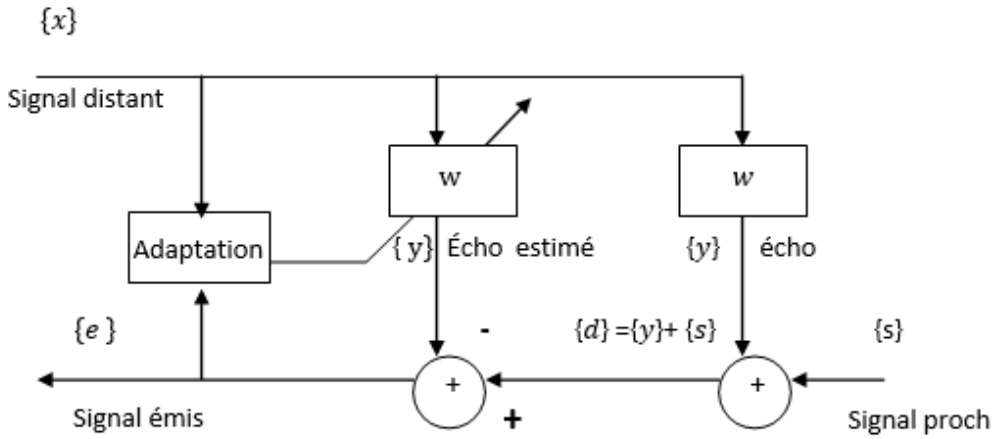


Figure I.2 *Structure complète d'un système d'annulation d'écho.*

I.5. Rôle du filtre adaptatif

Le rôle d'un filtre adaptatif est d'ajuster le paramètre w pour un objectif bien défini (minimisation de l'EQM : erreur quadratique moyenne) [2]

I.6. Applications du filtrage adaptatif

Le filtrage adaptatif trouve ses applications dans divers domaines, comme le traitement de signal, les communications numériques et le contrôle automatique. Ci-dessous, on présente quatre grands axes d'application [3]

- L'identification de systèmes ;
- La prédiction ;
- La modélisation inverse ;
- L'annulation d'interférences.

I.6.1. Identification de systèmes

Le but d'une telle configuration, est la détermination d'un modèle du système à identifier. Quand *e(n)* sera nulle, on aura les paramètres du système qui seront égaux à ceux du filtre adaptatif (cf. la Figure (I-3)) [3].

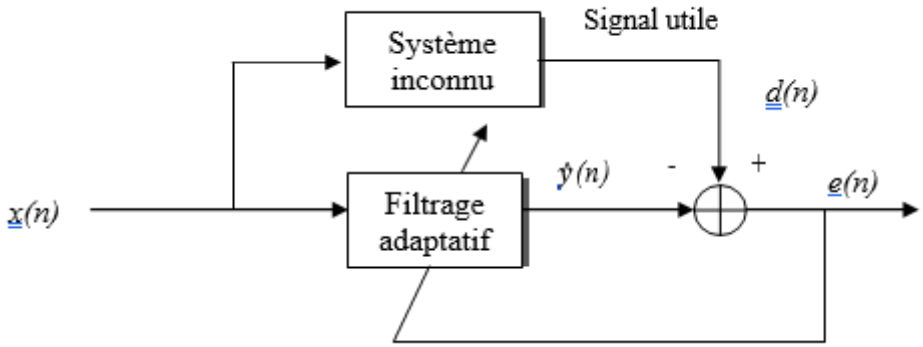


Figure I.3 *Identification du système.*

I.6.2. Prédiction [3]

Dans un tel système, on veut prédire la sortie donnée par $\hat{y}(n)$ en utilisant le vecteur d'entrée $X(n) = [x(n)\ x\ (n-1), \ldots\ x(n-L+1)]^T$, où $(\cdot)^T$ désigne l'opérateur de la transposée.

Ce filtre, (Figure I-4), présente deux sorties ; $\hat{y}(n)$, qui est la valeur prédite et fortement corrélée avec *x(n)*. Cette méthode peut être utilisée dans des applications d'amélioration de lignes. L'erreur
de prédiction *e(n)*, qui n'est pas corrélée avec *x(n)*, est utilisée dans différentes applications comme le Codage Prédictif Linéaire (LPC).

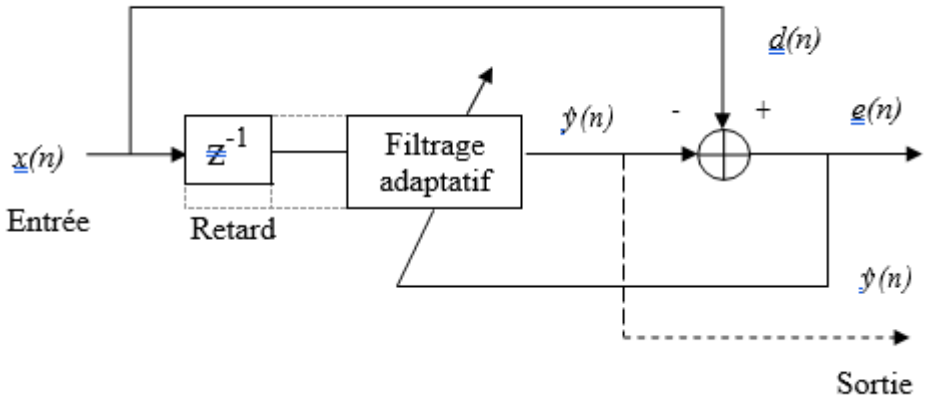


Figure I.4 *Filtre prédictif direct d'erreur.*

I.6.3. Modélisation inverse

Le schéma ci-dessous (Figure I.5) représente le principe de base de la modélisation inverse utilisée pour réduire la dispersion et éliminer l'interférence entre symboles dans les systèmes de communications numériques rapides, comme dans les modems [3]

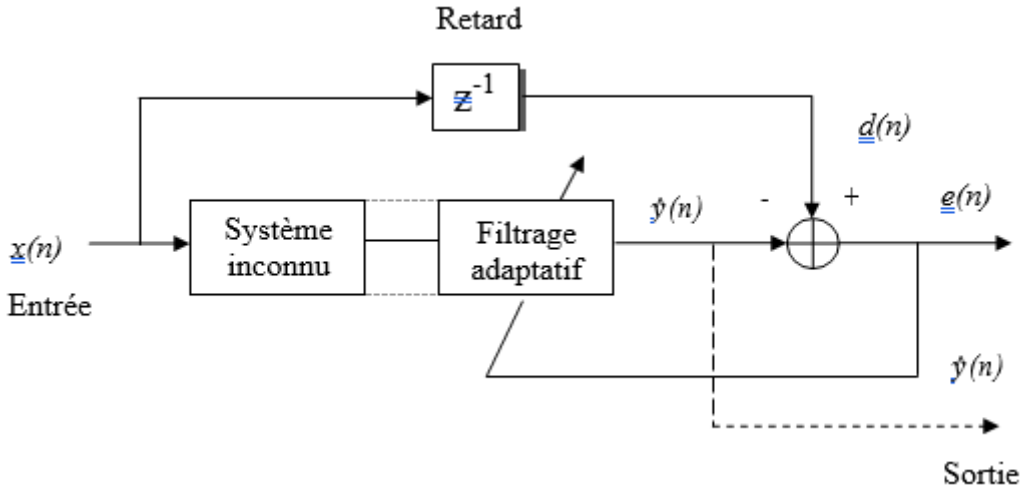


Figure I.5 *Modélisation inverse.*

I.6.4. Annulation d'interférences [3]

La Figure (I.6), montre le schéma bloc d'un filtre adaptatif, le signal primaire comporte les interférences à éliminer. Un signal *x(n)* avec un bruit 1, $b_1(n)$ avec lequel il est décorrélé forment le signal primaire du filtre (signal primaire = $x(n) + b_1(n)$).
Le signal de référence bruit 2, $b_2(n)$, est supposé être décorrélé avec le signal x(n), sinon, on aura la dégradation du rapport signal sur bruit SNR (Signal to Noise Ratio). Le signal bruit 2 est filtré pour donner une sortie, $\hat{y}(n)$, qui se rapproche le plus possible du signal bruit 1. Cette sortie est soustraite du signal primaire $d(n)=x(n)+b_1(n)$, pour donner la sortie du système
$e(n)=x(n) + b_1(n) - \hat{y}(n)$.

De plus, le signal de référence est dénué (ou presque) d'information et est fourni par un capteur proche de celui de d(n).

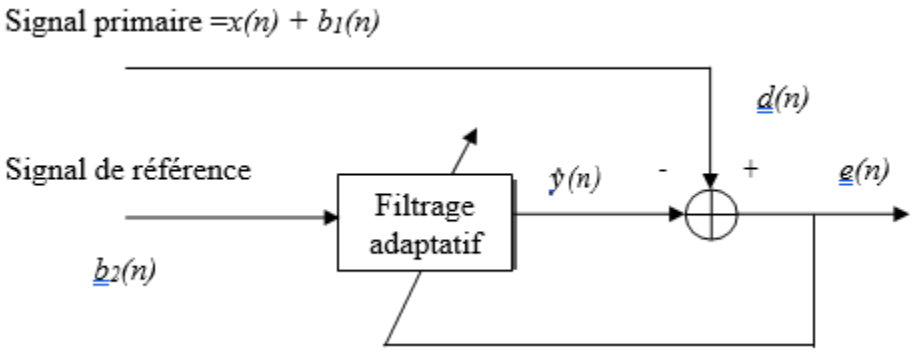


Figure I.6 *Annulation d'interférences*

I.7. Du filtrage de Wiener au filtrage adaptatif

Il existe deux classes importantes de filtres linéaires optimaux [3] :

- Filtrage de Wiener (où les signaux considérés *d(n)* et *x(n)* sont stationnaires)
- Filtrage de Kalman (qui est une généralisation du filtre de Wiener valable aussi dans lecas de processus (où de signaux) non stationnaires).

I.7.1. Filtrage de Wiener

Le filtrage de Wiener doit son nom au scientifique Norbert Wiener, est adéquat pour les situations dans lesquelles le signal ou le bruit sont stationnaires. Considérons la figure (I.7) suivante

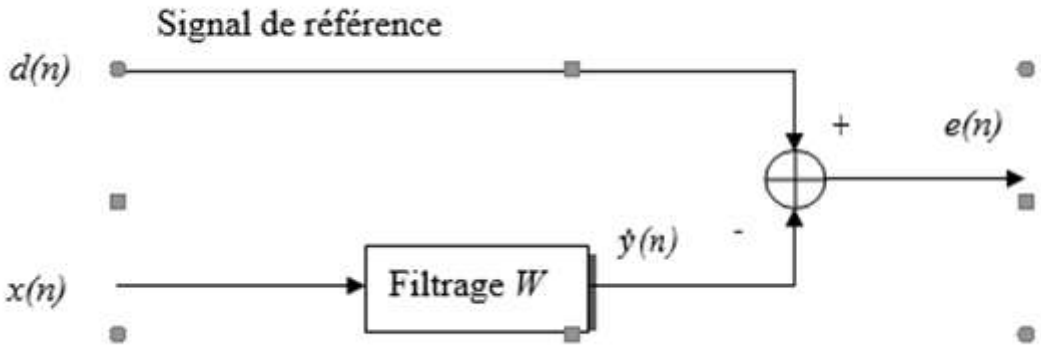


Figure I.7 *Schéma du filtrage de Wiene*r**.**

Le problème du filtre optimal est de trouver le « meilleur » filtre, c'est-à-dire celui permettant d'obtenir en sortie une réponse *ŷ(n)* la plus « proche » possible d'une réponse désiré *d(n),* lorsque l'entrée est une certaine séquence *x(n).*

Le problème consiste donc à rechercher le filtre assurant l'erreur la plus faible *e(n)*, au sens d'un critère d'optimisation **[3]**

$$W_{opt} = argmin_w J(e(n)) \quad (I.1)$$

$$W_{opt} = [w_{opt,0} \quad w_{opt,1}, \dots \quad w_{opt,-1}] \quad (I.2)$$

Où W_{opt} est le vecteur optimum, et $\{ J\}$ représente la fonction de coût :

$$J = \{e^2(n)\}$$

De nombreux choix sont possibles en ce qui concerne la fonction de coût comme indiqué précédemment. L'erreur quadratique moyenne (MSE) est la plus utilisée, car elle conduit àdes développements mathématiques complets et simples, fournissant une solution unique en fonction des caractéristiques au second ordre des variables aléatoires. C'est sur l'estimation linéaire en moyenne quadratique que repose le filtrage de Wiener.

La sortie du filtre s'écrit :

$$\hat{y}(n) = \sum_{l=0}^{L} x(n-l)\, w_l \quad (I.3)$$

I.7.2. Principe d'orthogonalité

Le vecteur optimum W_{opt} est celui qui annule le gradient du critère :

$$J = \{e^2(n)\}$$

$$e(n) = d(n) - X^T(n)\, W(n) \quad (I.4)$$

$$\frac{6J}{6W} = O_{L\times 1} \quad (I.5)$$

On a

$$\frac{\partial J}{\partial W} = 2E\{e(n)\frac{\partial J}{\partial W}\}$$

$$\frac{6J}{6W} = -2E\{e(n)\, X(n)\} \quad (I.6)$$

Par conséquent à l'optimum on a :

$$\mathrm{E}\{e_{min}(n)\mathrm{X}(n)\} = 0_{L\times 1} \quad \text{(I.7)}$$

Où $e_{min}(n)$ est l'erreur pour laquelle J est minimisée (c.-à-d pour le filtre optimal), c'est le principe d'orthogonalité signifiant que toutes les entrées x(n-l), tel que $0 < l < L-1$ sont décarrelées de l'erreur $e_{min}(n)$ [3]

En d'autre terme, le critère *J* est atteint si et seulement si l'erreur *e(n)* est orthogonale aux échantillons du signal *x (n-l)*.

I.7.3. Equation de Wiener – Hopf

En développant l'équation (I.7) nous obtenons :

$$E\{\mathrm{X}(n)(d(n) - \mathrm{X}^T(n)\mathrm{W}_{opt})\} = 0$$

Soit :

$$E\{\mathrm{X}(n)\, d(n)\} - E\{\mathrm{X}(n)\mathrm{X}^T(n)\}\,\mathrm{W}_{opt} = 0$$
$$E\{\mathrm{X}(n)\, d(n)\} = E\{\mathrm{X}(n)\mathrm{X}^T(n)\}\,\mathrm{W}_{opt}$$

Donc :

$$\mathrm{W}_{opt} = R_{L_{xx}}^{-1} P_{Lxd} \quad \text{(I.8)}$$

$R_{L,xx}= E\ \{X^T(n)\ X(n)\}$ est la matrice d'autocorrélation d'ordre *L* du signal d'entrée *x(n)*, cette matrice est définie positive, de Toeplitz et symétrique.

$P_{L,xd}= E\ \{X\ (n)\ d(n)\}$ est le vecteur d'intercorrélation entre la sortie désirée *d(n)* et l'entrée *x(n)*.

L'équation (I.8) est appelée l'équation de Wiener-Hopf.

L'équation de Wiener-Hopf assure l'existence et l'unicité de la solution optimale cependant, dans la plupart des applications pratiques, les statistiques des signaux sont inconnues a priori

où les signaux sont souvent non stationnaire, par conséquent, les statistique doivent être évaluées à partir des données recueillies aux entrées.

Ce type de problème amène à développer des méthodes de filtrage adaptatif où le filtre W change en fonction des informations reçues. L'erreur en sortie est utilisée pour la mise à jour des coefficients du filtre, qui sont ajustés à chaque itération par un algorithme d'adaptation, qui est une procédure itérative bien connue qui annule le gradient de l'erreur quadratique moyenne [3]

I.8. Définition d'un algorithme

La notion d'algorithme est souvent associée à l'informatique, pourtant le terme algorithme vient du nom du mathématicien perse du 9e siècle Muhammad Ibn Mūsā al-Khuwārizmī. La notion d'algorithme est donc très antérieure à la création du premier ordinateur [2].

Un algorithme est un énoncé d'une suite d'opération permettant de donner la réponse à un problème. Un algorithme est une spécification d'un schéma de calcul sous forme d'une suite finie d'opération élémentaire obéissant à un enchainement déterminé [2].

I.9. Le choix de l'algorithme

Le choix de l'algorithme se fera en fonction des critères suivant **[2]** :

- La rapidité de convergence qui sera le nombre d'itérations nécessaires pour converger « assez près » de la solution optimale de Wiener dans les cas stationnaire,
- La mesure de la proximité entre cette solution optimale et la solution obtenue,
- La capacité de poursuite (tracKing) des variations (non stationnarité) du processus on examinera quels sont les algorithmes vraiment adaptatifs.
- La robustesse au bruit ou mauvais conditionnement des données.
- La complexité ou le nombre d'opérations par itération + place mémoire nécessaire (mémoire de données).
- La structure ou aspect hardware, complexité de l'implantation matérielle.
- Les propriétés numériques (stabilité -précision) dans le cas d'une précision limitée sur les données et les coefficients

I.10. Algorithme du moindre carré moyen LMS [2]

On 1959, Widow et Hoff ont obtenu un algorithme dont le nom était Least Mean Square **« LMS »** et jusqu'à présent, il est l'un des meilleurs algorithmes de filtrage adaptatif. Cet algorithme est utilisé largement pour différentes applications telles que l'égalisation des canaux, annulation d'écho, suppression du bruit…etc.

Le **LMS** ajuste les coefficients de $w(n)$ d'un filtre pour réduire l'erreur quadratique moyenne entre le signal souhaité et la sortie du filtre. Le **LMS** est essentiellement le type de filtre adaptatif car pour converger vers la solution optimale de Wiener, il utilise le vecteur de

gradient de filtre. L'algorithme est également utilisé en raison de sa simplicité de calcul.

L'équation ci-dessous est l'algorithme LMS pour la mise à jour des poids du filtre adaptatif pour chaque itération.

$$W(n+1) = w(n) + \mu(n)x^*(n) \quad (I.9)$$

x (n): Vecteur des valeurs d'entrée temporisées ;
W (n) : Poids du vecteur au temps n ;
μ : C'est un paramètre qui contrôle le changement immédiat du facteur de mise à jour. Ce paramètre induit un grand impact sur la performance de l'algorithme LMS afin de changer sa valeur. Si la valeur de μ est trop petite, le filtre adaptatif prend beaucoup de temps à converger vers la solution optimale et en cas de grande valeur le filtre adaptatif sera divergent et devient instable.

I.11. Dérivation et l'algorithme LMS [2]

La dérivation de l'algorithme LMS est le développement de la méthode de la plus grande ponte et s'inspire de la solution optimale de Wiener. Cet algorithme utilise fondamentalement les formules qui met à jour les coefficients du filtre en utilisant les vecteurs poids w et également mettre à jour le gradient de la fonction de coût en fonction du coefficient de pondération du filtre $\boldsymbol{\varepsilon}(n)$.

L'algorithme LMS est très simple, il nécessite seulement (2L + 1) multiplications et 2 L additions par itération, ou L est le nombre de coefficients du filtre.

$$W(n+1) = w(n) - \mu \nabla \boldsymbol{\varepsilon}(n) \quad (I.10)$$

Avec

$$\nabla \boldsymbol{\varepsilon}(n) = \nabla \{|e(n)|^2\} = E\{\nabla |e(n)|^2 = E\{e(n)\nabla e^*(n)\} \quad (I.11)$$

On a

$$\nabla e^*(n) = -x^*(n) \quad (I.12)$$

Ainsi on obtient

$$\nabla(n) = -\nabla E\{e(n)x^*(n)\} \quad (I.13)$$

En pratique, la valeur de l'espérance $E\{e(n)x^*(n)\}$ est normalement inconnue. Donc nous avons besoin d'introduire l'approximation ou estimée comme la moyenne de l'échantillon.

$$\hat{E}\{e(n)\, x^*(n)\} = \frac{1}{L}\sum_{i=0}^{L-1} e(n-l)x^*(n-l) \quad (I.14)$$

Avec cette estimation, on obtient le vecteur poids de mise à jour comme suit :

$$\mathrm{W}(n+1) = w(n) + \frac{\mu}{L}\sum_{i=0}^{l-1} e(n-l)x^*(n-l) \qquad \text{(I.15)}$$

Si nous utilisons une moyenne d'échantillon ponctuel (L=1) alors

$$E\{e(n)\mathrm{x}^*(n)\} = (n)\mathrm{x}^*(n) \qquad \text{(I.16)}$$

Et enfin, l'équation de mise à jour μ vectrice de poids devient la forme simple :

$$\mathrm{W}(n+1) = \mathrm{w}(n) + \mu(n)\mathrm{x}^*(n) \qquad \text{(I.17)}$$

$$0<\mu<\frac{1}{tr(R)}$$

I.12. Algorithme de moindres carrés normalisés (NLMS)

En utilisant le paramètre μ de taille normalisé dans le LMS, nous retrouvons face à un autre cas d'algorithme connu sous le nom du LMS normalisé (NLMS). La mesure pour calculer le vecteur de pondération de mise à jour est [2] :

$$(n)=\frac{\beta}{c+\|x(n)\|^2} \qquad \text{(I.18)}$$

Où :

$\mu(n)$: Est le paramètre de taille d'étape à l'échantillon n.

β : Le pas d'adaptation normalisé ($0< \beta < 2$)

c : Est une constante pour éviter la division par zéro lorsque la norme du vecteur x(n) est nulle.

I.13. Dérivation de l'algorithme NLMS

Le LMS normalisé (NLMS) est l'ajustement du LMS. La nécessité de dériver le NLMS est que la puissance du signal d'entrée change dans le temps et en raison de ce changement le pas entre deux coefficients adjacents du filtre change également et affecte le taux de convergence. Lors de petits signaux ce taux de convergence va ralentir et lors de hauts signaux ce taux de convergence va augmenter et induire plus grande erreur.

Pour contourner ce problème, on va essayer d'ajuster ce pas µ en permanence en fonction de l'énergie instantanée du signal d'entrée .Par conséquent, ce paramètre est dit normalisé.
On peut montrer que l'algorithme est stable si [2] :

$$0< \mu < \frac{2}{\lambda_{max}} Et\ 0 < \mu < \frac{2}{trace(R_x)} \quad (I.19)$$

λ_{max} Est le maximum des valeurs propres de la matrice d'autocorrélation de vecteur $x(n)$. Cependant, l'auto-corrélation R_x est généralement inconnue, pour cette raison, le maximum λ_{max} et R_x sont estimées pour utiliser les bornes. Pour résoudre ce problème, on introduit une nouvelle estimation de trace (R_x) comme :

$$trac(R_x)=(p+1)E\{|x(n)|^2\} \quad (I.20)$$

Pour $p = 0,1,\ldots$

$E\ \{|x(n)|\hat{}2\}$: est la puissance du signal d'entrée. Il peut être estimé par l'estimateur:

$$\hat{E}\ \{|x(n)\ |\ ^2\}= \frac{1}{P+1}\sum_{k=0}^{p}|x(n-k)|^2 \quad (I.21)$$

Par conséquent, les limites du paramètre deviendront :

$$0< \mu < \frac{2}{(P+1)E\{|x(n)|^2\}} \quad (I.22)$$

Par substitution l'équation (I.24) dans l'équation (I.25) on obtient le paramètre μ comme suit :

$$0< \mu < \frac{2}{x^*(n)x(n)} \quad (I.23)$$

Pour les processus complexes variables dans le temps, on calcule le paramètre detaille de l'échelon dans le temps (échantillon n) :

$$\mu = \frac{2}{x^H(n)x(n)} = \frac{\beta}{||x(n)||^2} \quad (I.24)$$

β : Le pas d'adaptation normalisé ($0 < \beta < 2$)

En remplaçant μ par μ(n) dans l'équation (I.17) pour mettre à jour le vecteur de poids dans algorithme LMS, nous atteignons un nouvel algorithme qui est le NLMS. La mise à jour du vecteur de poids est maintenant :

$$W(n+1) = w(n) + \frac{\beta}{||x(n)||^2} e(n)x^*(n) \quad (I.25)$$

Dans l'algorithme LMS, comme le vecteur de poids w(n) varie en fonction du signal d'entrée x(n), alors lorsque x(n) est très petit le calcul de poids de l'équation de mise à jour du vecteur sera très grand ce qui constitue un problème. Pour cette raison, on met en œuvre la constante « c » qui est un paramètre de régularisation.

$$\mathrm{W}(n+1) = w(n) + \frac{\beta}{c+||x(n)||^2} e(n)x^*(n) \qquad \text{(I.26)}$$

μ (n) : Est le gain d'adaptation.

L'algorithme NLMS est résumé comme suit :

ALGORITHME NLMS

Entrée

Initialisation du vecteur $\underline{W}(n) = 0$;
Vecteur d'entrée x(n) ;
Signal mesuré $d(n)$;
Le pas d'adaptation initial β ;
Constant $c = 0.000006$;
Longueur du filtre L ;

Sortie

Sortie u filtre $\hat{d}$(n) ;
Coefficients optimaux $\underline{W}(n)$;

Procédure

$\hat{d}$(n) =$W^T . d(n) = x^T(n) . \underline{w}(n)$
$e(n) = d(n) - \hat{d}$(n)
$\underline{w}(n+1) = w(n) + \frac{\beta}{c + \|x(n)\|^2} e(n)\, x^*(n)$

I.14.Algorithmes NLMS proportionnel

L'annulation de l'écho est un domaine de recherche actif depuis plusieurs décennies. Récemment, ce problème est confronté à plus de défis en raison du développement d'applications grand écran centrées sur le réseau telles que la voix sur IP (VOIP). L'augmentation du délai de transmission du réseau et la complexité des programmes d'application introduisent plus de retard dans le chemin vocal, ce qui se traduit par un temps de retour d'écho plus long. Pour annuler l'écho efficacement, nous devons utiliser des filtres adaptatifs plus longs pour modéliser le chemin d'écho réel, ce qui augmentera la complexité de calcul et diminuera considérablement la vitesse de convergence des algorithmes adaptatifs La plupart des algorithmes adaptatifs populaires pour traiter ce problème sont des algorithmes Normalized Least Mean Square (NLMS) en raison de leur simplicité et leur robustesse. Étant donné que l'algorithme NLMS a une vitesse de convergence lente pour les entrées couleur et que les longs filtres adaptatifs peut avoir une grande erreur de réglage, une faible vitesse de convergence et une demande de calcul élevée, plusieurs variantes de cet algorithme a été proposé pour améliorer ses performances [2]

L'algorithme de mise à jour partielle est un moyen efficace de réduire la complexité du calcul en ne mettant à jour que certains coefficients à chaque période d'échantillonnage. Une caractéristique très importante de la trajectoire d'écho dont nous voulons profiter pour augmenter les performances de l'algorithme de filtrage adaptatif est la parcimonie. Si nous pouvons localiser les grands coefficients et seulement mettre à jour ces coefficients, nous adaptons en fait un filtre beaucoup plus court, augmentant ainsi la vitesse de convergence et diminuant la complexité de calcul. Malheureusement, en général, nous n'avons pas cette information a priori, alors nous devons chercher les gros coefficients. Les algorithmes bien connus qui tirent parti de la rareté du chemin d'écho sont le NLMS proportionnel (PNLMS) [2]

I.14.1. Algorithme PNLMS

L'algorithme PNLMS a été proposé pour la première fois en introduisant une matrice diagonale de commande pas à pas $G\ (n)$ pour associer un paramètre pas à chaque coefficient. Cette technique consiste à adapter chaque étape à une valeur désirée avec un taux de convergence qui varie d'une étape d'adaptation à l'autre. Cet algorithme exploite la faible densité des réponses impulsionnelle pour réaliser une adaptation sensiblement plus rapide que celle réalisée par l'algorithme conventionnel NLMS. L'algorithme PNLMS nécessite plus d'opérations que l'algorithme NLMS mais il présente l'avantage de converger plus rapidement que ce dernier. Une version normalisée peut être obtenue en normalisant l'équation de mise à jour avec la norme euclidienne du vecteur d'entrée afin d'éliminer l'influence de la puissance du signal d'entrée .Avantage du PNLMS est qu'il nécessite moins d'informations a priori sur le chemin d'écho, puisque la mise à jour ne dépend que des estimations de filtre actuelles l'algorithme PNLMS est donné par [2] :

$$\gamma_{mi}(n) = \rho max\ \{\delta_p, |w_0(n)|, |w_1(n)|, \dots \dots |w_{L-1}(n)|\} \quad \text{(I.27)}$$

$$\gamma(n) = max\{\gamma_{min}(n), |w_l(n)|\}, \quad 0 \leq l < L \quad \text{(I.28)}$$

$$g_l(n) = \frac{y_l(n)}{(1/L)\sum_{i=0}^{L-1} y_i(n)} \quad 0< l < L \quad \text{(I.29)}$$

$$G(n) = diag[g_0(n) \dots \dots g_{L-1}(n)] \quad \text{(I.30)}$$

$$W(n-1) = w_i(n) + \frac{\mu G(n)x(n-i)e(n)}{x^T G(n)x(n)+\delta_{PNLMS}} \qquad 0 \leq i \leq N\text{-}1 \quad \text{(I.31)}$$

$$-1<\delta_p<1$$

Où δ_{PNLMS} est la constante de régularisation, L est la longueur du filtre adaptatif et $diag\ \{g_0(n)\ \dots\ \dots\ g_{L-1}(n)\}$ une matrice diagonale avec les éléments$\{g_0(n) \dots \dots g_{L-1}(n)\}$

Les valeurs typiques des constantes d'algorithme sont données comme suit :

$$\delta_{PNLMS} = 0.01 \text{ et } \rho = 5/L$$

I.14.2. Algorithme PNLMS amélioré (IPNLMS)

Les algorithmes adaptatifs proportionnels ont été initialement conçus pour une réponse impulsionnelle sparse. Pour une réponse impulsionnelle non clairsemée (non-sparse), la vitesse de convergence de l'algorithme PNLMS se dégrade considérablement, même pire que l'algorithme NLMS. Le PNLMS amélioré (IPNLMS) est proposé pour éviter la dégradation dans le cas où l'impulsion sous-jacente la réponse est non-sparse. La raison pour laquelle l'algorithme PNLMS converge plus lentement que l'algorithme NLMS pour réponse impulsionnelle dispersive est en raison du choix brutal, l'opérateur maximum. Au lieu de ce choix brutal, la moyenne des coefficients actuels est adoptée pour éliminer partiellement les effets négatifs d'une estimation inexacte de w (n). Par conséquent, l'algorithme IPNLMS converge aussi rapidement comme l'algorithme PNLMS pour la réponse impulsionnelle sparse et sa performance n'est pas pire que l'algorithme NLMS pour un dispersive [2]

I.15. L'algorithme RLS

L'algorithme RLS implémente récursivement une solution de moindres carrés exacte. Comme indiqué précédemment, la solution Wiener pour un filtre adaptatif de longueur finie est donnée par :

$$w_{opt} = R^{-1}p \tag{I.32}$$

Où R est la matrice d'autocorrélation des entrées et p est la corrélation entre les entrées et le signal de référence. À chaque intervalle de temps, RLS estime récurremment R^{-1}et p sur la base de toutes les données antérieures et calcule le vecteur de poids comme :

$$w_n \stackrel{\text{def}}{=} R^{-1}p_n \tag{I.33}$$

Ce qui est donc la meilleure approximation à jour de la solution Wiener. La formule de mise à jourde poids peut être écrite comme suit : [1]

$$w_n = w_{n-1} + \mu_n R_{n-1}^{-1} a_n X_n \tag{I.34}$$

Où

$$a_n = d(n) - w_{n-1}^T X_n \tag{I.35}$$

$$\mu_n = \frac{\beta^{-1}}{1+\beta^{-1}X_n X_n^T R_{n-1}^{-1}} \tag{I.36}$$

β : est une constante légèrement inférieure à une.

Cette formulation met en évidence l'opération de décorrélation réalisée par RLS sur les données d'entrée : le gradient stochastique $\alpha_n X_n$ est pré-multiplié par une estimation de la matrice d'autocorrélation inverse, qui a pour effet de décorréler les entrées du filtre adaptatif. Cela réduit la sensibilité de l'algorithme à sa diffusion de valeur propre d'entrée. La prémultiplication par R^{-1} peut malheureusement nuire à la stabilité du filtre si la matrice R est mal conditionnée ; Une situation qui se produit chaque fois que le filtre contient plus de poids que nécessaire [2].

RLS est un algorithme adaptatif qui tente de minimiser l'erreur carrée moyenne (MSE). C'est une version mise à jour du filtre Wiener bien connu. Il a un taux de convergence rapide vers la solution optimale, des performances élevées pour les signaux non stationnaires et un minimum de MSE minimum en convergence. Ces attributs le rendent adapté à l'amélioration de la parole, à l'égalisation des canaux et aux applications d'annulation d'écho [2].

I.15.1. Dérivation de l'algorithme RLS

Au moment n, toutes les valeurs de l'erreur d'estimation depuis le début de l'algorithme RLS sont requises. De toute évidence, au fur et à mesure que le temps progresse, la quantité de données requises pour traiter cet algorithme augmente. Le fait que la mémoire et les capacités de calcul sont limitées rend l'algorithme RLS une impossibilité pratique dans sa forme la plus pure. Cependant, la dérivation suppose toujours que toutes les valeurs de données sont traitées. En pratique, seul, un nombre fini de valeurs précédentes est considéré, ce nombre correspond à l'ordre du filtre RLS RIF, La dérivation suivante du RLS est résumée de Farhang-Boroujeny [1].$y_n\ (k)$ est défini comme la sortie du filtre RIF, à n, en utilisant le vecteur de poids et le vecteur d'entrée d'un temps précédent k. La valeur d'erreur d'estimation en $e(k)$ est la différence entre la valeur de sortie souhaitée à l'instant k et les valeurs correspondantes de $y_n(k)$. Pour k = 1,2, 3,..., n**.**

$$y_n\,(k)=w^T(n)X(n) \tag{I.37}$$

$$e_n\,(k)=d(k)-y_n(k) \tag{I.38}$$

$$d\ (n) = [d\ (1), (2),....d(n)]^T \tag{I.39}$$

$$(n) = [y_n\ (1),y_n\ (2)....y_n(n)]^T \tag{I .40}$$

$$e\ (n) = [e(1),e_n(2)....e_n(n)]^T \tag{I.41}$$

$$e\ (n)= d(n)-y(n) \tag{I.42}$$

I.15.2.Mise en œuvre de l'algorithme RLS

Comme indiqué précédemment, la mémoire de l'algorithme RLS se limite à un nombre fini devaleurs, correspondant à l'ordre du vecteur de poids du robinet de filtre. Tout d'abord, il convient de noter deux facteurs de la mise en œuvre du RLS: le premier est que bien que l'inversion de la matrice soit essentielle à la dérivation de l'algorithme RLS, aucun calcul d'inversion matricielle n'est nécessaire pour la mise en œuvre, réduisant ainsi considérablement la complexité computationnelle de l'algorithme. Deuxièmement, contrairement aux algorithmes basés sur LMS, les variables actuelles sont mises à jour au cours de l'itération qu'ils doivent utiliser, en utilisant les valeurs de l'itération précédente [1].

I.16.Critères de comparaison entre les algorithmes adaptatifs

Dans la comparaison des performances entre les algorithmes d'identification et de filtrage adaptatifs, plusieurs critères de performances peuvent être utilisés pour évaluer objectivement les performances des nouvelles versions algorithmiques proposées dans la littérature et celle déjà publiées. On distingue deux sortes de critères. La première concerne les performances intrinsèques de l'algorithme, c'est-à-dire sa vitesse de convergence, sa capacité de poursuite et sa robustesse au bruit sur la sortie. La seconde vient des contraintes externes de l'application, englobant d'une part la complexité arithmétique (nombre d'opérations et mémoire requise), la stabilité numérique, et le retard de traitement pour les algorithmes par blocs. Ces critères et d'autres sont résumés comme suite [3]

- Erreur en régime permanent ;
- Stabilité numérique ;
- Vitesse de convergence en phase initiale ;
- Vitesse de convergence en régime permanent ;
- Influence de la taille du système sur les performances des algorithmes ;
- Influence de la parole comme signal d'excitation ;
- Influence du bruit additif sur le signal écho ;
- Complexité de calcul.

I.17. Conclusion

Nous avons conclu le filtre adaptatif est un filtre qui adapte ses coefficients au cour du temps en fonctionde l'évolution des caractéristiques statistiques du signal non stationnaires d'entrée.

Il est utilisé chaque fois pour d'identification de processus, pour la prédiction de signal ou pour supprimer des perturbations situées dans le domaine de fréquences du signal utile, ce que les filtres classiques ne peuvent pas les assumer [15].

Un filtre adaptatif est constitué de deux parties distinctes :

- ✓ un filtre numérique à coefficients ajustables
- ✓ un algorithme de modification des coefficients basé sur un critère d'optimisation

Ainsi, dans ce chapitre, nous avons fait un rappel des théories du filtrage adaptatif, en commençant par le filtrage adaptatif, puis le filtrage du Wiener, et enfin quelque techniques pour certains algorithmes.

Chapitre II:Les réseaux de neurones artificiel

RNA

II.1 Introduction

Partant d'un souhait initial de mieux comprendre la performance du cerveau humain, les réseaux de neurones artificiels sont une classe de modèles qui ont eu un impact significatif dans le domaine de l'analyse et du traitement des données. Cependant, l'inspiration pour les réseaux neuronaux vient de la volonté de créer des systèmes artificiels sophistiqués, une vision intelligente, capables d'effectuer des opérations similaires à celles effectuées régulièrement par le cerveau humain.

Les réseaux de neurones comprennent actuellement différents domaines tels que la médecine, l'électronique, l'informatique, l'automatisation, la robotique, la classification, le contrôle, le traitement du signal, le traitement d'image, etc.

Dans ce chapitre nous présentons un rappel théorique sur Réseaux de neurones (Neurone biologique, Neurone formel) et réseau de neurones artificiels (Définition, fonctionnement de RNAs, historique de RNAs, perceptron multicouche, Réseaux de neurones à fonction de base radiale (RBF)), théorème d'approximation universelle, avantages et inconvénients des réseaux de neurones.

II.2 Réseaux de neurones [4]

Au fil des années et à l'ère de la robotique, les chercheurs ont toujours cherché à inverser certains aspects de l'intelligence humaine sur les machines. Pour cela, un modèle mathématique du cerveau humain a été conçu ce qui a donné la naissance de réseaux neuronal artificiel originellement dérivé du modèle biologique. De leurs caractéristiques :

- La capacité d'apprendre.
- La possibilité de sauvegarder des informations.
- La capacité à traiter des informations incomplètes.

II.2.1 Neurone biologique

Le cerveau humain est composé d'un réseau de neurones très compliqué, dont, il contient. Plus que 10^{10} [5] cellules nerveuse connectées entre eux. Le neurone est une cellule Composée d'un corps cellulaire et d'un noyau (Figure 2.1). Le corps cellulaire se ramifie pour former ce que l'on nomme les dendrites. Celles-ci sont parfois si nombreuses que l'on parle alors de chevelure dendritique ou d'arborisation dendritique [4].

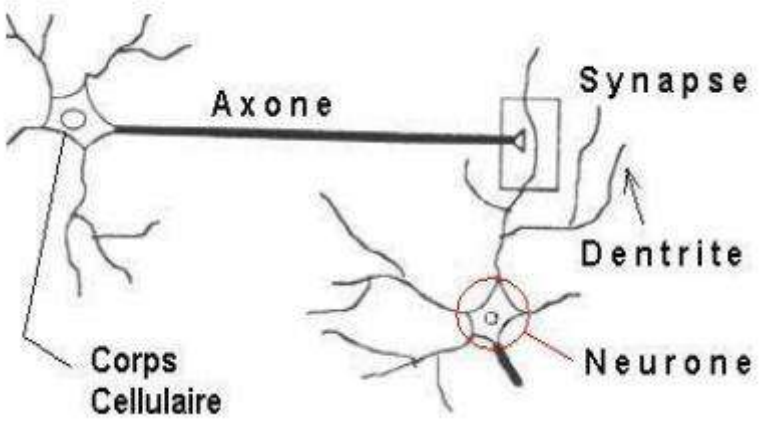


Figure II.1 *Schéma d'un neurone biologique*

II.2.1.1 Structure de neurone

La structure d'un neurone peut être décomposée en quatre parties distinctes [6] :

- **Les dendrites :** forment un maillage de récepteurs nerveux qui permettent d'acheminer vers le corps du neurone des signaux électriques en provenance d'autres neurones.
- **Le corps cellulaire :** c'est le noyau du neurone, il fait la sommation des signaux qu'il reçoit puis faire passer la somme a un seuillage.
- **L'axone :** c'est le moyen de transport des signaux émis par le corps cellulaire vers un autre neurone.
- **Synapse :** La transmission entre deux neurones n'est pas directe. En fait, il existe un espace intercellulaire de quelques dizaines d'Angströms (10-9 m) entre l'axone du neurone afférent et les dendrites (on dit une dendrite) du neurone efférent. La jonction entre deux neurones est appelée la **synapse.**

Basé sur les concepts : dendrites, corps cellulaire, axone, synapse, et le principe de transmission des signaux, les chercheurs ont imaginé un modèle similaire de neurones qui sont des neurones formels

II.2.2 Neurone formel

C'est un processeur très simple imitant grossièrement la structure et le fonctionnement d'un neurone biologique, la première version du neural formel est celle de Mc.culloche et Pitts qui est largement utilisée [4]. Ce dernier, est un processus élémentaire (automate) à *n* entrées et une seule sortie. Chaque entrée est associée à un poids synaptiques *V* qui représente la force de la connexion

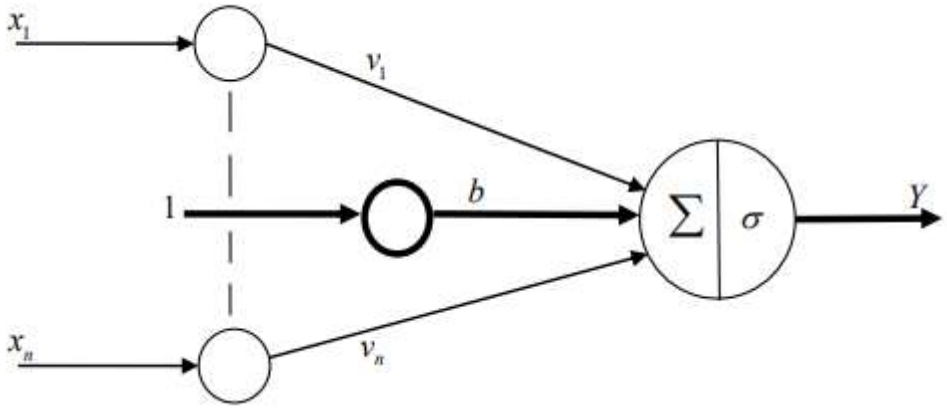


Figure II.2 *Neurone formel.*

Le neurone formel est composé de trois parties comme suit :

II.2.2.1 Poids

Les coefficients v_i sont les poids associés à chaque entrée x_i .Dont, chaque entrée est multiplié par le poids qui lui correspond est le résultat sera ($V^T X$) avec : V= $[v_1 \cdots v_n]^T$et X= $[x_1 \cdots x_n]^T$.

II.2.2.2 Seuil

Le biais b présente le seuil de neurone formel, ce paramètre additionnel est considéré comme le poids de l'entrée unité.

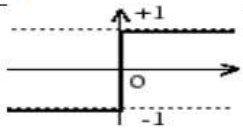
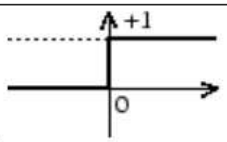
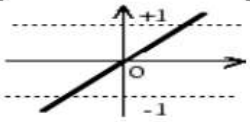
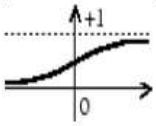
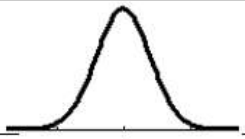
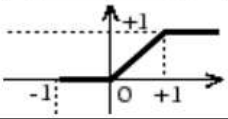
II.2.2.3 Fonction d'activation

La fonction d'activation calcul l'état du neurone, cette valeur sera transmise aux neurones avals. Il existe de nombreuses formes possibles de la fonction d'activation, les plus courantes sont présentées dans le tableau (II.1) [4] [5]

Comme conclusion, le résultat de cet automate s'exprime par un simple produit scalaire entre le vecteur d'entrées (X) et le vecteur des poids (V), ensuit le seuillage par un biais b et la fonction d'activation σ sont appliqués au résultat obtenu et la sortie finale du neurone sera [4] :

$$Y= \sigma(V^T X + b)$$

Tableau II.1 : *Fonction d'activation*

Fonction d'activation	Modèle mathématique	Graphe
Fonction signe	$\sigma(x)=\begin{cases}1 & si \quad x\geq 0\\ -1 & si \quad x\leq 0\end{cases}$	
Fonction seuil	$\sigma(x)=\begin{cases}1 & si \quad x>s\\ 0 & si \quad x\leq s\end{cases}$	
Fonction linéaire	$\sigma(x)=x$	
Fonction sigmoïde	$\sigma(x)=\frac{1}{1+\exp(-x)}$	
Fonction gaussienne	$\sigma(x)=\frac{1}{\sqrt{2\pi\sigma'}}e^{\frac{(x-c)^2}{2\sigma'^2}}$	
Fonction saturation	$\sigma(x)=\begin{cases}+1 & si \quad x\geq +1\\ x & si \quad -1\leq x\leq +1\\ -1 & si \quad x\leq -1\end{cases}$	

On remarquera que le neurone biologique a une sortie de nature binaire, donc le neurone peut prendre seulement deux état (active ou désactive).

La plupart des fonctions d'activation continues, offrant une infinité de valeurs possibles comprissent dans l'intervalle $[O,+1](ou[-1,+1])$

Enfin, nous pouvons dire qu'un ensemble de neurones formels liés les uns aux autres est un réseau de neurones artificiels.

II .3 Réseau de neurones artificiels

II.3.1 Définition

Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau [7]. Beaucoup de topologie sont été développées, parmi eux : multicouche, à connexion locale, à connexion récurrentes, connexion complète..., etc.

II.3.2 Fonctionnement de RNAs

Le fonctionnement d'un RNA est défini par la forme de connexion entre ses éléments. Par l'ajustement des pondérations, le réseau neuronal artificiel peut effectuer plusieurs tâches, de la même manière que le cerveau humain tel que : la reconnaissance de forme, la modélisation et la commande des processus,…etc. Autrement dit, la reconnaissance des tâches est enregistré par le changement dans les pondérations, cette phase est appelée apprentissage [8].

II .3.3 Historique de RNAs

Le développement des RNAs se résume comme suit :

- En 1949 décrits dans son ouvrage "The Organization of Behaviour". Hebb a proposé une règle simple qui permet de modifier la valeur des coefficients synaptiques en fonction de règle de Hebb» est presque partout présent dans les modèles actuels, même les plus sophistiqués .

-En 1958, *F. Rosenblatt* propose le premier réseau de neurone artificiel pratique pour la reconnaissance des formes, c'est le Perceptron. Ce modèle est composé de deux couches, la couche d'entrée et la couche de sortie [8] comme il est présenté par la figure (II.3).

Avec :σest la fonction seuil (modèle de F.Rosenblatt) X= $[x_1 \cdots x_ne]^T \in \Re^{ne}$ est le vecteur d'entrée, Y= $[y_1 \cdots y_2]^T \in \Re^{ns}$ est la matrice des poids et *ne* et $n_s$sont le nombre des neurones dans la couche d'entrée et de sortie respectivement.

La loi d'apprentissage de perceptron est inspirée de la règle de D. O. Hebb :' En cas d'erreur en sortie le vecteur d'entrée correspondant est ajouté ou retranché au vecteur des poids synaptiques'.

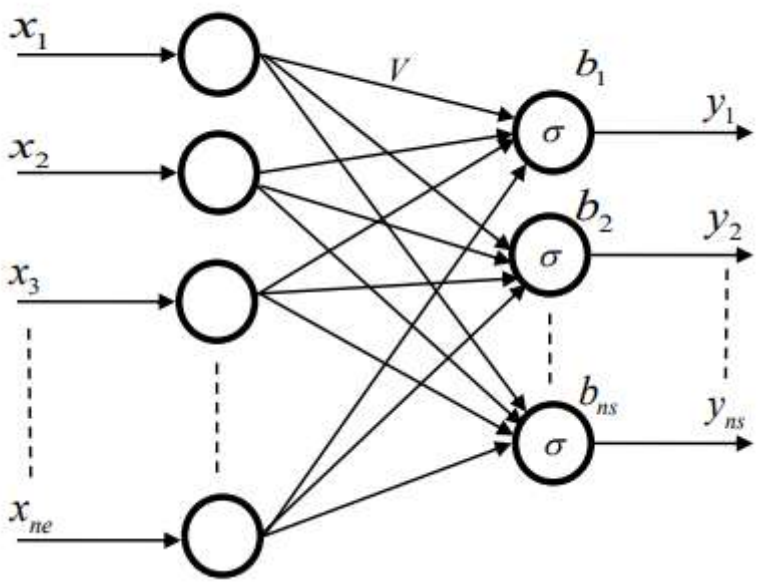


Figure II .3 *Perceptron à deux couches.*

- De leur côté, en 1960 Widrow est proposé l'Adaline (**Ada**ptative **Lin**ear**E**lement) : le premier réseau de neurones destiné à résoudre un problème du monde réel, la suppression d'échos sur une ligne téléphonique [9].

- 1969 : Après des débuts prometteurs, les réseaux de neurones subissent la concurrence des techniques de calculs plus classiques et tombent en désuétude suite notamment aux critiques de Minsky et Paper. Ceux-ci mettent en évidence l'incapacité des réseaux à une couche à calculer l'opérateur logique 'XOR' et les difficultés théoriques posées par l'apprentissage dans les réseaux multicouches [9].

II.3.4 Perceptron multicouche

En 1986 grâce au travail de *D. Rumelhart*, Une révolution survient dans le domaine des réseaux de neurones artificiels: une nouvelle génération de réseaux de neurones, le perceptron multicouche. Ce dernier est composé d'une couche d'entrée, une couche de sortie et une/où plusieurs couches cachées, comme il est illustré sur la figure (II.4). Ce réseaux est capable de traite tous les problèmes linéairement séparable ou non. Et il est considéré comme le réseau de contrôle le plus applicable.

Le perceptron multicouche est un réseau orienté de neurones artificiels organisé en couches et ou l'information voyage dans un seul sens, de la couche d'entrée vers la couche de sortie. Son apprentissage est de type supervisé, par correction des erreurs.

Dans ce cas uniquement, le signale d'erreur est « rétro propagé » .La fonction d'activation σ est située dans les couches cachées [4].

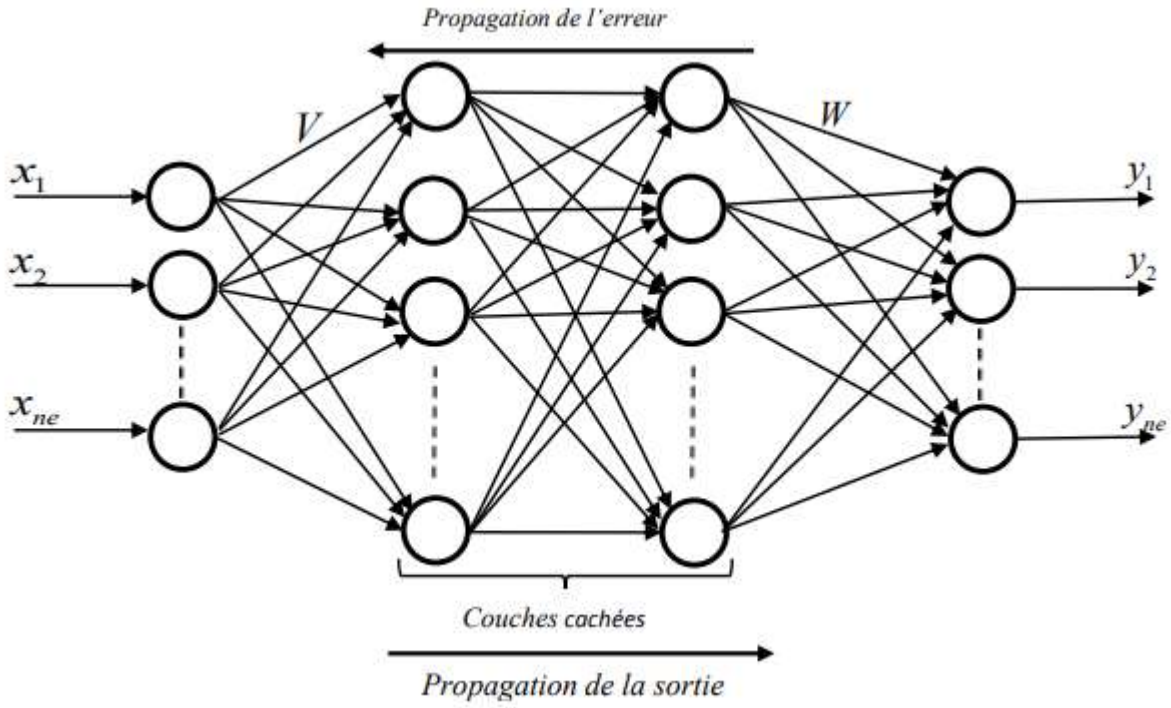


Figure II.4 *perceptron multicouche.*

II.3.4.1 Algorithme de rétro-propagation

L'algorithme de rétro-propagation est basé sur la méthode de gradient descendant [5], dont, l'apprentissage est supervisé. L'ajustement des poids se fait dans le sens inverse de la couche de sortie vers la couche d'entrée d'où l'appellation rétro-propagation (back propagation). Au début, la sortie du réseau est calculée et comparée avec la sortie désirée, ensuite l'erreur résultante est utilisée pour l'apprentissage des poids du la couche de sortie et du la dernière couche cachée. De la même manière, l'erreur du réseau se propage pour l'ajustement des poids des autres couches cachées jusqu'à la couche d'entrée.

L'opération d'apprentissage supervisé d'un réseau de neurones artificiels est résumée par la figure (II.5) [4],

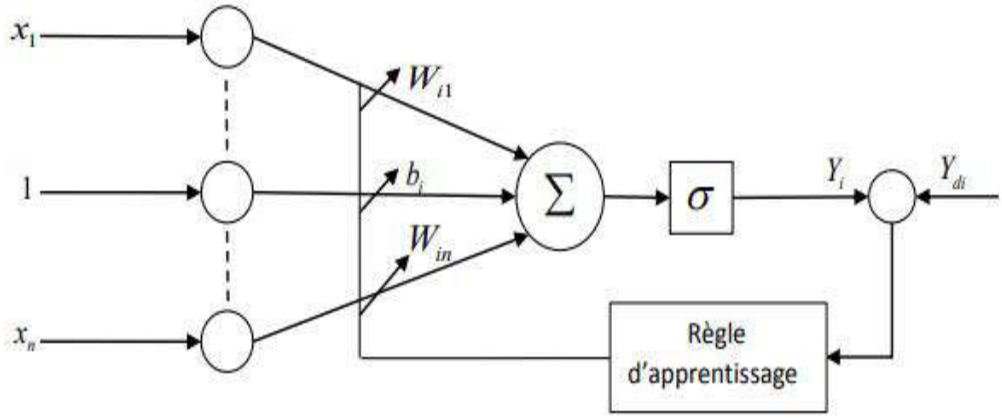


Figure II .5 *Mécanisme d'apprentissage.*

II.3.4.1. a. Algorithme de gradient

L'algorithme de gradient est basé sur la minimisation de la fonction de coût, pour l'apprentissage sous la supervision de réseaux neuronaux artificiels Cette fonction est donnée par [4] :

$$j_i = \frac{1}{2}E_i^2 = \frac{1}{2}(y_{di} \cdots y_i)^2 \qquad \text{(II.1)}$$

Avec :

E_i: est l'erreur du neurone de sortie i , y_{di} : est la sortie désirée de neurone de sortie i

y_i: est la sortie réelle de neurone de sortie i.

L'algorithme du gradient pour l'ajustement des poids et de biais est donnée par :

$$\begin{cases} w_{ij}(t) = w_{ij}(t) - \mu \frac{\partial J_i}{\partial w_{ij}(t)} \\ b_i(t+1) = b_i(t) - \mu \frac{\partial J_i}{\partial b_i(t)} \end{cases} \qquad \text{(II.2)}$$

La constante $0 < \mu < 1$représente le gain d'apprentissage.

II.3.4.1.b. Étapes de rétro-propagation

La sortie d'un neurone artificiel multicouche se calcul de la couche d'entrée vers la couche de sortie. Par contre, la minimisation de l'erreur entre la sortie réelle et la sortie désirée du RNA par la méthode de gradient est obtenue par la propagation du signale d'erreur dans le

sens inverse c'est-à-dire de la couche de sortie vers la couche d'entrée. Alors, l'ajustement des poids se fait dans le même sens qui le signale d'erreur et l'algorithme de gradient qui réalise cette tâche est appelé la rétro-propagation [8].

Les différentes étapes à suivre lors de l'apprentissage d'un réseau de neurones à propagation avant avec cet algorithme sont comme suit [4] :

- **Étape 1** : Initialiser les poids *Wi111j* et les seuils internes des neurones à des petites valeurs aléatoires.
- **Étape 2 :** Présenter le vecteur d'entrée et de sortie désirée pour l'apprentissage.
- **Étape 3 :** calculer le signale de sortie du RNA comme suit :

- La sortie de chaque neurone *j* de la couche cachée k est donnée par :

$$O_j^k = \sigma\left(\sum_{i=1}^{nc_{k-1}} w_{ij}^{k-1} O_i^{k-1}\right), \begin{cases} k = \overline{1 \div p,} \\ j = \overline{1 \div nc_k} \end{cases} \quad \text{(III.3)}$$

Avec :σ est la fonction d'activation, w_{ij}^{k-1} sont les poids liant la couche k−1 et la couche k , nc_{k-1} est le nombre de neurones dans la couche cachée k−1 , p est le nombre de couches neurones dans la couche d'entrée et $O_i^0 = x_i$ désigne l'entrée du RNA.

- La sortie de chaque neurone de sortie *l,* est donnée par :

$$y_l = \sum_{i=1}^{nc_p} w_{ij}^p o_i^p \,, \begin{cases} l = \overline{1 \div ns} \\ j = \overline{1 \div l} \end{cases} \quad \text{(II.4)}$$

Avec *ns* est le nombre de neurones dans la couche de sortie.

- **Étape 4 :** Calculer :

- les erreurs de chaque neurone de la couche de sortie l par l'expression suivant : *k*

$$e_l = y_{dl} - \sum_{i=1}^{nc_p} w_{ij}^p o_i^p \,, \begin{cases} l = \overline{1 \div ns} \\ j = \overline{1 \div l} \end{cases} \quad \text{(II.5)}$$

Avec y_{dl} est la sortie désirée de neurone de sortie *l.*

- les erreurs de chaque neurone caché *j* dans la couche caché *k* par l'expression suivant :

$$e_j^k = \sum_{i=1}^{nc_{k+1}} w_{ij}^{k+1} e_i^{k+1}, \begin{cases} k = \overline{p \div 1}, \\ j = \overline{1 \div nc_k} \end{cases} \quad \text{(II.6)}$$

Avec est e_i^{k+1} l'erreur e_i définie dans (II.6).

- **Étape 5 :** En utilisant l'algorithme de gradient, les poids est ajusté de la couche de sortie à la couche d'entrée comme suit:

$$w_{ij}^{k-1}(t+1) = w_{ij}^{k-1}(t) + \mu e_j^k \sigma'\left(\sum_{i=1}^{nc_{k-1}} w_{ij}^{k-1}(t)\, O_i^{k-1}\right) O_i^{k-1}, \begin{cases} k = \overline{p+1 \div 1} \\ i = \overline{1 \div nc_{k-1}} \\ j = \overline{1 \div nc_k} \end{cases} \quad \text{(II.7)}$$

Avec : $\sigma'(x) = \frac{d\sigma(x)}{dx}$

- **Étape 6 :** refaire les étapes de 2 à 5 jusqu'à l'obtention d'une fonction cout proche de zéro (inférieur à la valeur désirée).

II.3.5 Réseaux de neurones à fonction de base radiale (RBF)

L'idée générale des réseaux RBF dérive de la théorie d'approximation des fonctions, ces réseaux sont une architecture Feedforward puissante. La raison de son application vient du fait que le réseau utilise des fonctions gaussiennes standard qui sont à symétrie radiale. Son apprentissage est basé sur l'algorithme K-means et l'algorithme des moindres carrées [4].

Les réseaux de neurones RBFs, sont principalement utilisés pour résoudre des problèmes d'approximation de fonctions dans des espaces de grandes dimensions. Ils sont lus adaptés, en raison d'apprentissage local. Ce type d'apprentissage peut rendre le processus d'entrainement bien plus rapide que dans le cas d'un perceptron multi-couche, qui apprend de façon globale. En effet, ces réseaux avec une seule couche cachée peuvent approximer n'importe quelle fonction continue ayant un nombre fini de discontinuités sur tout compact [4].

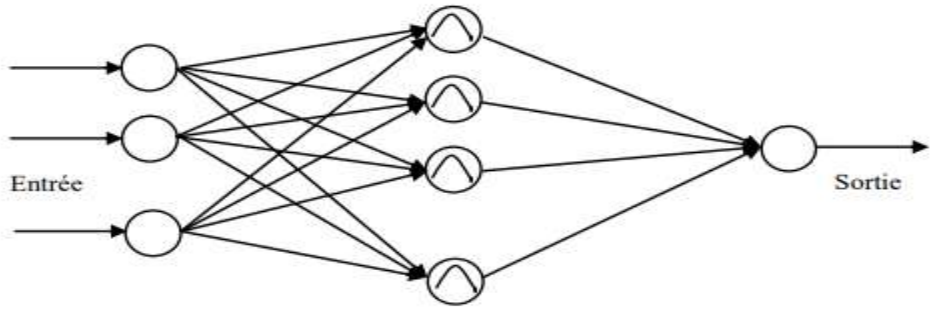


Figure II.6 *réseaux à fonction de base radiale.*

II.4 Théorème d'approximation universelle

Parmi les caractéristiques des réseaux de neurones, l'approximation universelle est la plus importante du point de vue de l'identification et la commande. L'approximation universelle a fait le sujet de beaucoup de travaux de recherche et plusieurs démonstrations pour des réseaux de neurones avec différentes fonction d'activation [4]. On dispose d'un nombre important de publications montrant que les réseaux de neurones (RA) sont bien des approximateurs universels, c'est-à-dire, pour toute fonction continue $f(x)$ défini sur un ensemble compact Ω, et pour toute constante positive ε, il existe un RN d'ordre supérieur, avec k neurones dans la couche cachée, telle que :

$$sup_{x\in\Omega}|f(x) - \hat{f}(x)\,| \leq \varepsilon \qquad \text{(II.8)}$$

Avec $\hat{f}(x)$ est l'approximation neuronale de $f(x)$ et ε est une constante positive définie l'erreur d'approximation neuronale [4].

II.5 Avantages et inconvénients des réseaux de neurones

II.5.1 Avantages

Les principales qualités des réseaux de neurones sont leur capacité d'adaptabilité et d'auto-organisation et la possibilité de résoudre des problèmes non-linéaires avec une bonne approximation [4]. Ils ont une bonne immunité aux bruits et se prêtent bien à une implantation parallèle. La rapidité d'exécution est une qualité importante et elle justifie souvent à elle seule le choix d'implanter un réseau de neurones. Ces qualités ont permis de réaliser avec succès, plusieurs applications : classification, filtrage, compression de données, contrôleur, etc.…

II.5.2 Inconvénients

La difficulté d'interpréter le comportement d'un réseau de neurones est un inconvénient pour la mise au point d'une application. Il est souvent impossible d'utiliser les résultats obtenus pour améliorer ce comportement. Il est également hasardeux de généraliser à partir

d'expériences antérieures et de conclure ou de créer des règles sur le fonctionnement et le comportement des réseaux de neurones [4].

Plusieurs paramètres doivent être ajustés et aucune méthode ne permet de choisir des valeurs optimales. Beaucoup d'heuristiques sont utilisées, mais elles se contredisent parfois et elles ne permettent pas toujours de trouver des valeurs optimales [4].

II.6 Conclusion

Dans ce chapitre, les principaux concepts de réseau de neurone inspiré par l'étude du cerveau humain ont été présentés, Où des modèles plus complexes ont été développés à travers le développement de la neurobiologie et l'utilisation d'outils théoriques plus puissants. Tels que l'algorithme de rétro-propagation. De plus, le modèle multicouche est le modèle de réseaux de neurones le plus utilisé.

Chapitre III : Filtres adaptatifs utilisant les réseaux de neurones

III.1. Introduction

Un réseau de neurones formels est un ensemble de cellules non linéaires élémentaires interconnectées. Chaque connexion est caractérisée par un coefficient ou poids synaptique, et le comportement du réseau dépend essentiellement des valeurs de ces poids ; l'apprentissage est l'opération par laquelle les poids synaptiques sont calculés de telle manière que le système réalise bien la fonction qu'on attend de lui. Dans tout ce qui suit, nous nous plaçons dans le cadre d'un apprentissage au cours duquel on cherche à calculer les poids de manière à satisfaire à un critère qui fait intervenir la différence entre la réponse du réseau et une réponse désirée. Il a été prouvé que les réseaux de neurones formels sont des approximateurs universels [10], en ce sens que toute fonction multivariable non linéaire suffisamment régulière peut être approchée, avec une précision fixée, par un réseau de neurones, pourvu que celui-ci possède une architecture et une taille appropriées, et qu'il soit soumis à un apprentissage efficace ; ces deux points - architecture et apprentissage - sont essentiels dans toute recherche sur les réseaux de neurones. Dans la mesure où l'approximation de fonctions et l'identification de signaux ou de systèmes sont au cœur des préoccupations du filtrage adaptatif et de celles de la commande adaptative, il est naturel de chercher à évaluer l'apport potentiel des réseaux de neurones dans ces domaines, notamment pour la réalisation d'opérations de filtrage non linéaire en traitement du signal.

Le cadre conceptuel dans lequel nous nous plaçons est très différent du cadre "classique" d'apprentissage et d'utilisation des réseaux de neurones ; dans le domaine de la classification, par exemple, l'apprentissage est effectué de manière *non adaptative* puisque l'ensemble d'apprentissage est connu à l'avance, de taille finie, et que l'ordre de présentation des exemples est arbitraire ; une fois l'apprentissage terminé, le réseau de neurones est muni des coefficients calculés durant la phase d'apprentissage, et il est utilisé sans modification de ces coefficients. Bien entendu, il est possible de traiter des signaux temporels de manière non adaptative : c'est ce qui a été fait dans la plupart des recherches concernant la reconnaissance de la parole par des réseaux de neurones, ainsi que la prédiction de séries temporelles par ces réseaux. C'est ainsi qu'ont été introduits les "Time Delay Neural Networks" (TDNN), qui ont une architecture de filtres transverses non linéaires mis en cascade, utilisés, par exemple, pour la reconnaissance de phonèmes [11] ; leur apprentissage n'est pas adaptatif.

Nous désignons par réseau de neurones adaptatif un réseau de neurones dont l'apprentissage est effectué en permanence, pendant son utilisation : ainsi, un prédicteur de parole reçoit en permanence un signal de parole échantillonné, et doit s'adapter en permanence aux caractéristiques du signal, lesquelles ne sont pas connues à l'avance et peuvent varier au cours du temps. L'apprentissage est donc effectué à partir d'un ensemble de données, éventuellement infini, dont l'ordre de présentation est imposé par la nature temporelle des informations traitées. Dans cette optique, nous montrons que les réseaux de neurones adaptatifs ne sont pas autre chose que des filtres adaptatifs non linéaires, qui peuvent être non bouclés (filtres transverses) ou bouclés (filtres récursifs).

Dans ce chapitre, nous aborderons les réseaux de neurones non bouclés utilisés comme filtres transverses et apprentissage des réseaux de neurones adaptatifs, équivalence des notions de récurrence temporelle et de dépliement spatial et algorithmes pour l'adaptation des réseaux bouclés

III.2 Les réseaux de neurones utilisés comme filtres non lineaires

III.2.1 Modèle de neurone formel [14]

Un neurone formel est une cellule élémentaire de calcul dont la structure est donnée par la Figure III. 1. On appelle sortie du neurone *i* la quantité

$$Z_i = f_i(v_i), \tag{III.1}$$

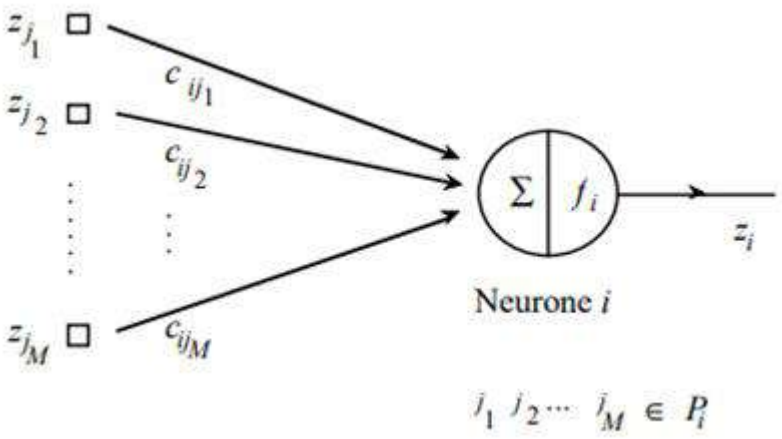


Figure III. 3 . *Neurone formel*

et potentiel du neurone la quantité

$$v_i = \sum_{j \in p_i} c_{ij} z_j \tag{III.2}$$

Où $\{c_{ij}\}$ est l'ensemble des poids synaptiques (ou des connexions), $\{z_j\}$ est l'ensemble des entrées du neurone *i* et f_i est une fonction non linéaire dite "fonction d'activation". P_i est l'ensemble des indices des entrées du neurone *i.* La fonction f_i peut être quelconque ; cependant, les auteurs utilisent généralement des fonctions impaires, croissantes, telles que des fonctions signe ou sigmoïdes. On supposera dans la suite que f_i est dérivable et bornée.

Il est clair qu'un filtre transverse obéissant à la relation (III.1) peut être réalisé par un "neurone linéaire" unique (f_i =identité), possédant M entrées dont les valeurs à l'instant n sont $\{u(n), u(n-1), \ldots, u(n-M+1)\}$, et dont la sortie est $y(n)$. On remarque également qu'un filtre récursif obéissant à la relation (III.3) peut être considéré comme un "neurone linéaire" unique, comportant M entrées externes $\{u(n), u(n-1), \ldots, u(n-M+1)\}$, et N entrées de bouclage $\{y(n\text{-}1), \ldots, y(n-N)\}$.

Pour réaliser des fonctions plus complexes de filtrage non linéaire, il est naturel d'envisager l'utilisation d'un *réseau* de neurones formels, c'est-à-dire d'un ensemble de cellules élémentaires définies ci-dessus, connectées entre elles : un neurone i reçoit des informations $\{z_j\}$ en provenance d'autres neurones et/ou d'entrées externes du réseau. L'architecture du réseau définit une relation entre les entrées externes et la sortie : les réseaux non bouclés permettent de réaliser des filtres transverses non linéaires, et les réseaux bouclés permettent de réaliser des filtres récursifs non linéaires.

III .2.2 Réseaux de neurones non bouclés utilisés comme filtre transverses

Dans un réseau non bouclé, l'information circule uniquement des entrées du réseau vers sa sortie: il n'y a pas de boucle de retour. Un réseau non bouclé dont les entrées à l'instant n sont les valeurs successives $u(n)$, $u(n-1)$, ..., $u(n-M+1)$ peut alors être considéré comme un filtre transverse non linéaire dont la sortie à l'instant n obéit à la relation :

$$y(n) = \Phi\,(u(n), u(n-1), \ldots, u(n-M+1) \qquad \text{(III.3)}$$

Où Φ représente la fonction non linéaire globalement réalisée par le réseau. On voit apparaître clairement l'intérêt potentiel des réseaux de neurones formels pour réaliser des opérations de filtrage, dans la mesure où un réseau de neurones peut, au moins en principe, approcher par apprentissage n'importe quelle fonctionnon linéaire suffisamment régulière.

L'architecture la plus générale d'un réseau de neurones utilisé comme filtre transverse est celle d'un réseau complètement connecté (Figure III.2). L'évolution d'un tel réseau, comprenant v neurones, M entrées externes et une sortie, dont le résultat global est exprimé par (III.3), est régie à l'instant n par les équations suivantes :

$$z_i(n) = u(n-i+1)\,; \qquad i = 1, \ldots, M,$$

$$z_i(n) = f_i(v_i(n))\,,$$

$$v_i(n) = \sum_{j<i} c_{ij} z_j\,(n); \qquad i = M+1, \ldots, M+\mathrm{v}\,,$$

$$y\,(n) = z_{M+\mathrm{v}}\,(n) \qquad \text{(III.4)}$$

Les entrées externes ou les sorties des neurones ont été indifféremment appelées $z_i(n)$ par leurs indices. En l'absence de contrainte particulière sur l'amplitude du signal de sortie du système, on choisit pour f_{M+v} la fonction identité.

L'architecture d'un réseau non bouclé, c'est à dire la topologie des connexions, peut être cmplètement ou partiellement imposée a priori par le problème à traiter : celui-ci impose en effet la suite des valeurs des signaux d'entrée et celle des sorties désirées, et, de surcroît, les connaissances a priori sur le problème peuvent donner des indications qui permettent de concevoir une architecture de réseau bien adaptée. Si cette architecture comporte des retards [12], il est toujours possible de représenter le réseau sous la forme canonique (III.3). Le passage d'une structure de réseau non bouclé quelconque à sa forme canonique, et en particulier la détermination de la taille *M* de la mémoire, sont détaillés dans [13]. Par exemple, il arrive que certaines applications en traitement du signal [14] imposent des structures de plusieurs filtres adaptatifs mis en cascade ; l'optimisation conjointe de tous ces systèmes mis en cascade peut alors être envisagée. La représentation d'une cascade de filtres transverses linéaires ou non linéaires par un réseau non bouclé multicouche est étudiée dans [14] et ne sera pas reprise ici. Remarquons simplement qu'une cascade de filtres transverses n'a d'intérêt que si les filtres sont non linéaires ou bien si le problème impose la structure en cascade. Dans le cas contraire, des filtres linéaires mis en cascade (ou réseau linéaire multicouche) sans que cela soit imposé par le problème, agissent simplement comme un seul filtre linéaire (ou réseau linéaire monocouche) adaptatif d'ordre plus élevé.

Si l'on ne dispose d'aucune connaissance a priori sur la structure à imposer au réseau, on peut utiliser une architecture de réseau non bouclé complètement connecté du type de celui de la Figure III 2.

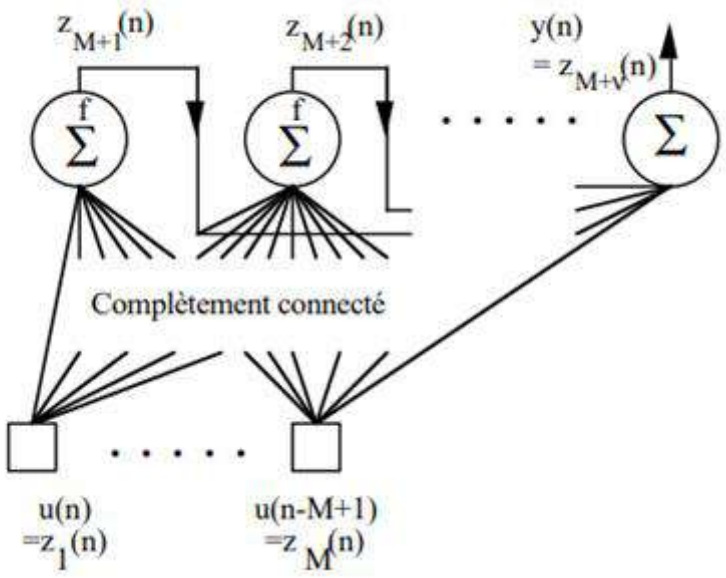


Figure III.4 . *Réseau de neurones non bouclé complètement connecté utilisé comme filtre transverse non linéaire.*

III.2.3 Réseaux de neurones bouclés utilisés comme filtres récursifs

Les filtres récursifs ont été introduits pour deux raisons essentielles : (i) la récursivité confère au filtre une mémoire infinie avec un nombre fini de coefficients ; (ii) pour modéliser des systèmes récursifs, il est naturel d'introduire des filtres récursifs. Ces deux mêmes raisons justifient l'utilisation de réseaux de neurones bouclés.
Les réseaux de neurones bouclés ont été largement étudiés en tant que mémoires associatives; dans ce cadre, le réseau est considéré comme un système dynamique non linéaire, dont les attracteurs sont mis à profit pour retrouver des informations mémorisées pendant l'apprentissage. Dans ce memoire, nous nous plaçons dans un contexte complètement différent: le réseau n'a aucune raison, en général, d'atteindre un état d'équilibre ou un cycle limite.

Représentation canonique d'un réseau de neurones bouclé

Comme dans le cas des réseaux non bouclés, il est commode d'introduire une représentation canonique de n'importe quel réseau bouclé dont l'architecture a pu être imposée préalablement par le problème à traiter [12]. La dynamique d'un réseau bouclé peut être décrite par une équation aux différences d'ordre N, qui peut être exprimée sous la forme de N équations aux différences du premier ordre mettant en jeu, outre les M entrées externes, N variables, dites variables d'état. Par conséquent, tout réseau bouclé peut être mis sous une forme canonique [13], constituée

-d'un réseau non bouclé dont les sorties à l'instant n sont composées de la sortie du réseau bouclé et dessorties d'un nombre N de neurones dits neurones d'état,

-et de N bouclages, de retard unité, reliant les sorties des neurones d'état aux entrées d'état correspondantes.

L'évolution du réseau bouclé est déterminée par les équations d'état suivantes :

$$y(n) = \psi[X(n),\ U(n)],$$

$$X(n+1) = \varphi[X(n),\ U(n)] \qquad \text{(III.5)}$$

où $U(n)$ est le vecteur des M dernières valeurs successives de l'entrée externe u et $X(n)$ est le vecteur d'état.
La sortie du réseau peut être aussi une variable d'état. Dans la suite, nous nous limiterons au cas où l'état est constitué de la sortie à l'instant n et de ses $N - 1$ valeurs précédentes (Figure III. 3) ce qui correspond au modèle NARMAX [14]. Les valeurs des entrées externes du réseau et des sorties des neurones sont les suivantes :

$i = 1, \ldots, M$; $z_i(n)$: valeurs des entrées externes,

$i = M + 1, \ldots, M + N$; $z_i(n)$: valeurs des composantes de l'état $X(n)$ à l'entrée du réseau,

$i = M + N + 1, \ldots, M + N + \nu - 1$; $z_i(n)$: valeurs des sorties des neurones cachés,

$i = M + N + \nu, \ldots, M + 2N + \nu - 1$; $z_i(n)$: valeurs des composantes de l'état $X(n+1)$ en sortie du réseau (la sortie $z_{M+N+\nu}(n)$ du système global est la première variable d'état).

Avec ces notations, le système d'équations (III.5), qui décrit l'évolution du réseau représenté sur la figure III. 3.

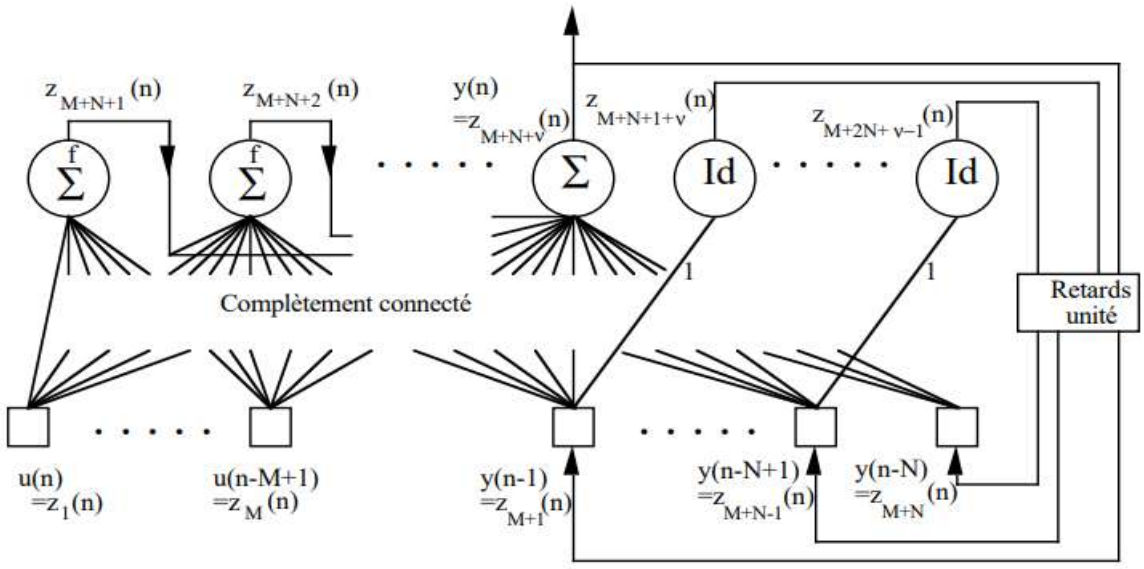


Figure III. 3. *Réseau de neurones bouclé (modèle NARMAX) utilisé comme filtre récursif non linéai*

Secret:

$$z_i(n) = u\,(n - i + 1)\,; \qquad i = 1, \ldots, M,$$

$$z_{M+i}(n) = z_{M+N+\nu+i-1}(n-1) \qquad i = 1, \ldots, N, \qquad \text{(III.6)}$$

$$z_{M+N+\nu+i-1}(n-1) i = 1, \cdots, N \qquad \text{(III.7)}$$

$$z_i(n) = f_i(v_i(n))\,,$$

$$v_i = \sum_{j<i} c_{ij} z_j\,(n); \qquad i = M + N + 1\,,\, M + N + \nu. \qquad \text{(III.8)}$$

Notons que la sortie du réseau à l'instant n est

$$Y(n) = z_{M+N+\nu}(n \qquad \text{(III.9)}$$

Cette écriture est conforme aux équations d'état (III.5), où le vecteur d'état $X(n)$ est composé des variables$Z_{M+N+v}(n)$ à $Z_{M+N}(n) = y(n-N)$, où le vecteur d'état à l'instant suivant $X(n+1)$ est composé des variables $Z_{M+N+v},(n)$ à $Z_{M+2N+v-1}(n)$le vecteur des entrées externes $U(n)$ est composé des variables $z_1(n)=u(n)$ à $z_M(n)=u(n-M+1)$, et enfin où φ et ψ sont des fonctions non linéaires qui dépendent à la fois des fonctions d'activation des neurones f_i et des coefficients c_{ij} . Remarquons que les neurones d'état, à l'exception du neurone de sortie, ne calculent pas leurs valeurs $Z_{M+N+v+1}(n)$ à $Z_{M+N+v+1}(n)$: ils ne font que transmettre ces valeurs suivant (III.7) ; grâce à ces notations, les algorithmes d'apprentissage introduits peuvent être présentés, et réalisés en logiciel, de manière complètement modulaire.

Si l'on ne dispose a priori d'aucune information sur la structure à adopter, il faut alors considérer directement le réseau le plus général possible. Le nombre v de neurones, les valeurs de M et N sont à déterminer en fonction du problème. Différentes architectures ont été considérées dans la littérature sur les réseaux de neurones [14], mais elles restreignent souvent le type de fonction φ et ψqui peuvent être réalisées.

III.3 Apprentissage des réseaux de neurones adaptatifs

A l'heure actuelle, l'apprentissage supervisé des réseaux de neurones est effectué, généralement, de manière non adaptative : l'ensemble d'apprentissage est de taille finie, connu à l'avance, et l'on distingue la phase d'apprentissage de la phase d'utilisation du réseau ; une fois l'apprentissage terminé, le réseau est utilisé avec les coefficients obtenus à l'issue de l'apprentissage. Si l'on veut modifier la tâche en cours d'utilisation du réseau, afin de prendre en considération de nouvelles informations, il faut arrêter la phase d'utilisation et reprendre la phase d'apprentissage. Or, pour qu'un réseau de neurones puisse réaliser des opérations de filtrage, il est essentiel qu'il puisse s'adapter en permanence à d'éventuelles évolutions des caractéristiques du signal, sans que son fonctionnement ne soit interrompu.

III. 3.1 Critère d'optimisation

Un réseau de neurones utilisé comme filtre fait correspondre à une séquence de données d'entrée $u(n)$, $u(n-1)$, …, une séquence de données de sortie $y(n)$, $y(n-1)$, …. La tâche qu'il doit réaliser est traduite par un critère d'optimisation. En principe, chaque application devrait posséder son propre critère : minimiser le taux d'erreursur les bitstransmis lors d'une communication numérique, maximiser le rapport signal sur bruit dans le cas d'un filtrage spatial, optimiser un critère d'écoute pour la prédiction de la parole, etc. Il se trouve que, dans beaucoup d'applications, le critère des moindres carrés est couramment utilisé parce qu'il conduit à des performances intéressantes et à des réalisations relativement simples. Ce critère est défini à partir d'une séquence d'entrée (éventuellement infinie) $u(n)$, $u(n-1)$, … et d'une séquence de réponses désirées correspondantes $d(n)$, $d(n-1)$, ….

Dans le cas où l'on dispose d'un nombre fini *K* de couples {*u*(*m*), *d*(*m*)}, le critère des moindres carrés (MC) se traduit par la minimisation de la fonction de coût

$$J_{MC}(C) \triangleq \frac{1}{K}\sum_{m=1}^{K} e(m)^2 \qquad \text{(III.10)}$$

$$e(m) = d(m) - y(m) \qquad \text{(III.11)}$$

Où *C* est le vecteur des coefficients à optimiser. C'est ce critère qui est couramment utilisé dans l'apprentissage supervisé des réseaux de neurones classifié : chaque couple correspond à un exemple *u*(*m*) et à la classe *d*(*m*) qui lui a été attribuée par le superviseur. Ce critère n'a de sens en *classification* que si la distribution des classes dans l'univers des formes est figée : les performances de généralisation du réseau (capacité à classer des formes qui n'ont pas été apprises durant la phase d'apprentissage) sont fonction de la représentativité de la distribution statistique des exemples présentés par rapport à l'ensemble de toutes les formes à classer. Ce critère n'a de sens en *filtrage* que si les séquences *u*(*m*) et *d*(*m*) sont stationnaires et si *K* est suffisamment grand pour représenter la statistique des séquences.

Comme nous l'avons mentionné plus haut, l'apprentissage d'un réseau ou d'un filtre reposant sur laminimisation de la fonction de coût (III.10) est non adaptatif.

Lorsque l'on désire réaliser un *apprentissage permanent*, on cherche à trouver, à l'instant *n*, un ensemble de coefficients *C*(*n*) tels que l'erreur à l'instant *n*+1 soit aussi petite que possible, en tenant compte de l'expérience passée, c'est-à-dire des propriétés statistiques des signaux d'entrée et des valeurs désirées correspondantes. Dans le cas où ces signaux sont stationnaires, les coefficients recherchés sont ceux qui minimisent la fonction

$$J_{MC}^{n}(C) \triangleq \frac{1}{n}\sum_{m=1}^{n} e(m)^2 \qquad \text{(III.12)}$$

Dans le cas où les signaux ne sont pas stationnaires, il n'est évidemment pas souhaitable de tenir compte de toute l'expérience passée pour modifier les coefficients du filtre ; pour calculer la modification des coefficients à l'instant n, on peut utiliser la fonction de coût

$$I\,(n) \triangleq \frac{1}{2}\sum_{m=n-N_c+1}^{n} e(m)^2 \qquad \text{(III.13)}$$

Où N_C correspond à un intervalle de temps qui est petit devant l'échelle de temps typique de stationnarité des signaux.

III.3.2 Algorithmes adaptatifs fondés sur l'estimation du gradient de la fonction de coût

La recherche d'algorithmes adaptatifs obéit essentiellement à trois motivations [14] :

La recherche analytique du minimum de la fonction de coût (III.13) par rapport aux coefficients du filtre est inextricable pour des systèmes non linéaires et/ou bouclés ; il est donc intéressant de chercher un algorithmeitératif de recherche de minimum de la fonction de coût ;

Pour limiter la complexité des calculs et les répartir dans le temps, il est intéressant de concevoir des algorithmes *récursifs* qui optimisent le système à l'instant *n* en fonction de ce qu'il était à un instant antérieur ;

Dans le cas non stationnaire, il est nécessaire d'adapter les coefficients du système, en fonction de l'évolution de l'environnement, au fur et à mesure que l'information arrive.

Les algorithmes utilisés habituellement pour l'apprentissage des réseaux de neurones tiennent compte essentiellement des deux premiers points ; le troisième point, en revanche, est rarement pris en considération. Ainsi, dans un contexte de traitement du signal par un réseau de neurones, un algorithme sera dit adaptatif s'il calcule, au fur et à mesure que le temps court, des modifications des coefficients du système à partir des informations passées; il est évidemment souhaitable que l'algorithme soit capable de poursuivre d'éventuels changements des caractéristiques du signal.

Dans sa forme la plus générale, une modification, à l'instant *n*, du vecteur des coefficients du système,est calculée itérativement suivant :

$$\Delta C(n) = C_{K_n}(n) - C_0(n) \qquad \text{(III.14)}$$

$$C_k(n) = C_{k-1}(n) + \mu_{k-1} D_{k-1}(n) \; ; \qquad \text{(III.15)}$$

Avec

$K = 1, \ldots, k_n$

Où K_n est le nombre d'itérations réalisées avant de modifier les coefficients; K_n peut dépendre de *n*. Dans lasuite, on ne s'intéressera qu'à l'algorithme du gradient où

$$D_k = -\frac{\partial I(n)}{\partial C}\Big|_{C=C_k(n)} \qquad \text{(III.16)}$$

et où μ_{K-1} est une suite de paramètres positifs, à déterminer, dont dépendent la vitesse de convergence, la stabilité, les capacités de poursuite de l'algorithme. Remarquons que d'autres algorithmes, par exemple les algorithmes de type "quasi-Newton", reposent également sur le calcul du gradient et peuvent donc entrer dans le cadre de ce qui suit.

Cet algorithme est récursif, car son initialisation, à l'instant *n*, tient compte des coefficients du système lorsde la dernière modification de ceux-ci

$$C_0(n) = C_{K_{n-T}}(n - T) \qquad \text{(III.17)}$$

Où *T* est l'intervalle de temps qui sépare deux modifications successives des coefficients.

Lors d'un apprentissage adaptatif, on veut pouvoir remettre à jour les coefficients même dans un cas non stationnaire : il faudrait donc, théoriquement, itérer l'algorithme du gradient (III.15) K_n fois pour un même couple de données $\{u(n), d(n)\}$, afin de converger vers le minimum (éventuellement local) de la fonction de coût $I(n)$. Ceci correspond au trajet **1** sur la figure III.4. Cependant, supposons qu'à l'instant $n + 1$, correspondant aux nouvelles données $\{u(n + 1), d(n + 1)\}$, la fonction de coût $I(n + 1)$ soit très proche de $I(n)$ (variations lentes de l'environnement) ; il n'est pas nécessaire d'itérer (III.15) plusieurs fois (voir trajet **2** sur la figure (III. 4).

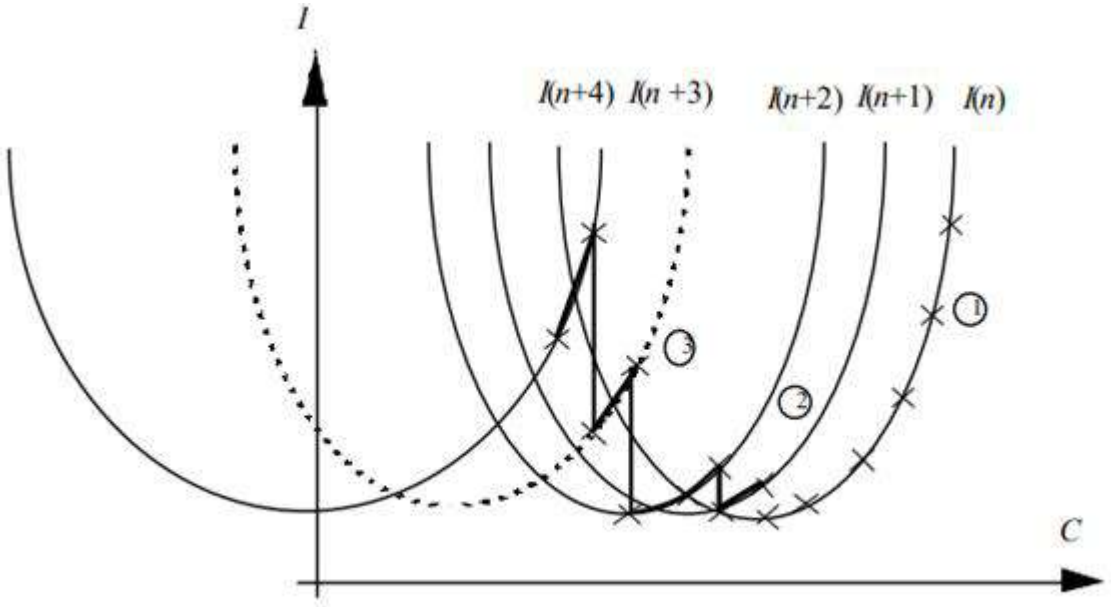


Figure III.4 *Surfaces d'erreur I(n).*

Par conséquent, dans un cas stationnaire, l'algorithme (III.14)-(III.15) peut se réécrire, en posant $K_n = 1$, $T = 1$, $D_0(n) = D(n - 1)$

Et

$$C_1(n) = C(n)$$

$$C(n) = C(n - 1) + \mu D(n - 1) \qquad \text{(III.18)}$$

Rappelons que, dans le cas où $N_c = 1$ dans la relation (IV.4), cet algorithme est connu sous le

nom de gradient stochastique.

Si, au contraire, $I(n + 3)$, par exemple (courbe en pointillé sur la figure(III .4), est très différente de $I(n + 2)$, une seule itération de (III.15) ne suffira pas à passer du minimum de $I(n + 2)$ au minimum de $I(n + 3)$ (voir trajet **3** sur la figure III. 4). Pour converger vers la solution optimale de $I(n + 3)$ appliquer la relation (III.18) plusieurs fois.

Cependant, dans un contexte non stationnaire où les fonctions de coût $I(n + 2)$, $I(n + 3)$ ... seraient successivement très différentes, on n'a pas forcément intérêt à obtenir successivement le minimum de chacune d'elles : on doit donc s'efforcer de réaliser un compromis entre la minimisation de la fonction de coût à chaque instant et la capacité à suivre les non-stationnarités. C'est là un problème délicat, auquel on est souvent confronté en filtrage adaptatif.

Le choix des paramètres N_c, T et K_n introduits dans le critère (III.13) et l'algorithme (III.14)-(III .17) est délicat et doit être fixé en fonction de connaissances a priori sur les caractéristiques des signaux à traiter (durée de stationnarité), de la tâche que doit réaliser le système, ainsi que du temps et des moyens dont on dispose pour effectuer les calculs. Si, par exemple, la modification des coefficients est faible, il n'est pas indispensable de prendre N_c plus grand que 1 dans la mesure où le moyen nage sera alors réalisé au cours des itérations successives de l'algorithme [14]; cependant, il faudra obligatoirement choisir T =1 pour que le moyen nage se fasse avec des valeurs de $C(n)$ assez voisines. Si au contraire, dans un cas stationnaire, on choisit $T > 1$, on aura intérêt à choisir $N_c > 1$ et éventuellement $K_n > 1$.

Dans un cas non stationnaire, N_c devra être choisi en fonction de la durée de stationnarité du signal. T sera choisi égal à 1 dans le cas où l'on a besoin d'adapter le système continuellement, mais on pourra envisager $T > 1$ dans le cas contraire. Dans le cas non stationnaire encore, on choisira K_n grand si l'on recherche à chaque fois une grande précision et si l'on a le temps de faire les calculs (or, souvent, qui dit contexte non stationnaire dit temps de calcul réduit). Au contraire, on choisira $K_n = 1$ dans le cas où l'on cherche à poursuivre un comportement moyen.

Une fois ces différents paramètres fixés, le problème essentiel réside dans l'évaluation du gradient afin de pouvoir implanter l'algorithme. Le filtrage adaptatif utilise couramment une technique de calcul du gradient que nous désignerons sous le terme de "calcul direct" ; l'apprentissage des réseaux de neurones met couramment en œuvre une technique de calcul du gradient connue sous le nom de "rétro propagation" [12] [14].

III.3.3 Evaluation du gradient dans un réseau non bouclé

III.3.3.1 Calcul direct

Le calcul direct du gradient repose sur le développement le plus immédiat de (III.13), (III.11), soit

$$\left.\frac{\partial I(n)}{\partial c_{pq}}\right|_{C=C_k(n)} = -\sum_{m=n-N_c+1}^{n} e(m) \left.\frac{\partial y(m)}{\partial c_{pq}}\right|_{C=C_k(n)} \qquad \text{(III.19)}$$

Dans cette expression, les erreurs $e(m)$ et les dérivées partielles sont calculées avec les derniers coefficients connus à l'instant n, $C_k(n)$; elles correspondent aux valeurs que l'on aurait calculées aux instants $m = n - N_C + 1, \ldots, n$ si l'on avait disposé de ces coefficients. Les dérivées partielles des $y(m)$ par rapport aux coefficients c_{pq} (dans le réseau complètement connecté mais non bouclé de la figure 2 on a $p > q$) se
Calculent à partir des équations d'évolution (III.4) du réseau. Tous ces calculs sont faits à l'instant n avec les coefficients $C_k(n)$; pour alléger l'écriture, nous omettrons dans la $suite\ la\ notation|_C$

Avec

$c = c_k(n)$

Il vient pour chaque $m \in [n - N_C + 1, n]$

$$\frac{\partial z_i(m)}{\partial c_{pq}} = 0; p > i \text{ ou } i = 1, \ldots, M,$$

$$\frac{\partial z_i(m)}{\partial c_{pq}} = f_i\big(v_i(m)\big) z_q(m); p = i \text{ et } i > M,$$

$$\frac{\partial z_i}{\partial c_{pq}} = f_i\big(v_i(m)\big) \sum_{p<j<i} c_{ij} \frac{\partial z_i(m)}{\partial c_{pq}}; \; p < i \text{ et } i > M. \qquad \text{(III.20)}$$

Ainsi, le calcul des $\frac{\partial y(m)}{\partial c_{pq}}$ se fait de proche en proche par propagation, depuis les entrées vers les sorties du réseau non bouclé linéaire dont les poids sont $f_i\big(v_i(m)\big) c_{ij}\big(cf.(IV.11)\big)$, des dérivées partielles des sorties des neurones situés en amont. Dans le cas où $N_c \geq 1$, il faut calculer les N_c dérivées partielles et termes d'erreurs intervenant dans (III.19) avec les poids fixés à leur valeur $C = C_k(n)$. On dit alors que N_c copies du réseau à l'instant n sont utilisées pour calculer les valeurs nécessaires au calcul du gradient (III .19) [13].

III.3.3.2 Calcul par rétro propagation

Le calcul du gradient par rétro propagation [14] fait intervenir des dérivées partielles différentes de cellesutilisées dans le calcul direct. A partir de (IV.4), on écrit

$$\left.\frac{\partial I(n)}{\partial c_{pq}}\right|_{C=C_k(n)} = \frac{1}{2}\sum_{m=n-N_c+1}^{n} \frac{\partial e^2(m)}{\partial v_p(m)} \left.\frac{\partial v_p(m)}{\partial c_{pq}}\right|_{C=C_k(n)} \quad \text{(III.21)}$$

Comme précédemment, nous omettrons dans la suite la notation $|C=C_k(n)$
D'après (III.4), on calcule aisément

$$\frac{\partial v_p(m)}{\partial c_{pq}} = z_q(m), p > M. \quad \text{(III.22)}$$

Les autres dérivées partielles intermédiaires intervenant dans (III.21) peuvent se calculer également à partirde (III.4) :

$$\frac{\partial e^2(m)}{\partial v_p(m)} = \sum_{1>P} \frac{\partial e^2(m) \partial v_l(m)}{\partial v_l(m) \partial v_p(m)} \quad \text{(III.23)}$$

Soit encore

$$\frac{\partial e^2(m)}{\partial v_p(m)} = f_p\left(v_p(m)\right) \sum_{1>P} c_{lp} \frac{\partial e^2(m)}{\partial v_l(m)} \quad \text{(III.24)}$$

La dernière relation exprime que les dérivées partielles$\frac{\partial e^2(m)}{\partial v_p(m)}$ peuvent être calculées à partir de celles déjà calculées en aval du réseau. C'est ce qu'on appelle le calcul du gradient par rétro propagation des dérivées partielles dans le réseau non bouclé linéaires dont les coefficients sontc_{lp} $f_p\left(v_p(m)\right)$ (II.24) [13]

On remarquera que la sommation se fait sur le premier indice *l* dans (III.24) alors qu'elle se faisait sur le deuxième indice dans le calcul direct (III.20) Dans un réseau non bouclé, les deux techniques de calcul du gradient présentées ci-dessus sont équivalentes du point de vue du résultat. Cependant, on a montré [12] que la rétro propagation et le calcul direct ont une complexité de calcul respectivement de l'ordre de v^2 et de v^4. En pratique, on utilisera donc la rétro propagation pour un réseau non bouclé.

III .4 Equivalence des notions de récurrence temporelle et de dépliement spatial

La représentation d'état d'un dispositif bouclé temporellement, qu'il soit linéaire ou non linéaire, comme celui de la figure III.3, représentation issue du formalisme utilisé en automatique, permet une interprétation "spatiale" particulièrement simple des calculs de gradients présentés dans la section précédente. En effet, l'erreur $e(m)$ est calculée, à l'instant courant n, par le dispositif, obtenu en "dépliant" spatialement la récurrence temporelle du réseau de la figure III .3. Ce dispositif est constitué d'une cascade de cellules élémentaires ou copies du réseau non bouclé de la forme canonique (§ III.3), contenant les mêmesvaleurs de coefficients $C_k(n)$ [14].

De cette façon, le calcul, à l'instant n, des erreurs $e(m)$ et des dérivées partielles de ces erreurs prend en considération un horizon temporel de longueur m ; ceci peut être interprété simplement comme le calcul de ces mêmes quantités sur une cascade non bouclée de m cellules identiques. C'est ce que nous appelons l'équivalence entre les notions de dépliement spatial et de récurrence temporelle [14].

III.5 Algorithmes pour l'adaptation des réseaux bouclés

III.5.1 Dépliement du réseau bouclé en un nombre fini de copies

Le calcul des erreurs et des dérivées partielles depuis l'instant initial n'est pas réalisable en pratique, car il nécessite un trop grand nombre de calculs, ainsi que le stockage en mémoire de tous les échantillons des signaux présentés au réseau ; de plus, dans le cas où les signaux sont non stationnaires, il n'est pas souhaitable de tenir compte de tout le passé. On est donc amené à tronquer les récurrences (III.6)-(III.9), et (III.25)-(III.28) ou (III.30)-(III .35) sur une longueur N_t . Autrement dit, on remplace la représentation développée, qui utilise m copies pour calculer l'erreur $e(m)$, par celle de la figure 6, qui utilise un nombre fixe de copies N_t ($N_t < m$) pour estimer $e(m)$. Pour calculer les N_c termes du gradient, il y a donc N_c systèmes de ce type. Le système d'argument m est un réseau non bouclé dont les entrées sont les entrées externes $Z_1^1(m), \dots, Z_M^1(m), Z_1^2(m), \dots, Z_M^2(m), \dots, Z_1^N(m), \dots, Z_M^{Nt}(m)$ ($Z_1^1(m)$ est la valeur de la i-ème entrée de la I-éme copie du m-ième système), les variables d'état de la première copie $Z_{M+1}^1(m), \dots, Z_{M+N}^1(m)$, et dont la sortie est $Z_{M+N+v}^{Nt}(m)$, laquelle constitue une estimation de $y(m)$. La modification des coefficients à l'instant n est la somme des modifications calculées sur chacun des N_c systèmes décrits ci-dessus [14].

Afin de diminuer le nombre de calculs, on peut utiliser une seule récurrence, c'est à dire un seul système composé de N_t copies qui calcule les N_c erreurs et les dérivées partielles. Les entrées de ce système unique sont les entrées externes $Z_1^1, \dots, Z_M^1, Z_1^2, \dots, Z_M^2, \dots, Z_1^{Nt}, \dots, Z_M^{2Nt}$, les variables d'état de la première copie Z_{M+1}^1 , Z_{M+N}^1, et les sorties sont $Z_{M+N+v}^{Nt-Nc+1}, \dots Z_{M+N+v}^{Nt}$. L'argument m, qui était le numéro de l'un des N_c systèmes, n'apparaît plus ici. Le résultat de ces calculs est évidemment différent de celui que l'on obtientavec les N_c systèmes indépendants [13].

III.5.2 Une nouvelle famille d'algorithmes adaptatifs [14]

Dans ce paragraphe, nous introduisons une nouvelle famille d'algorithmes adaptatifs qui s'appuient surles systèmes décrits dans le paragraphe précédent et qui résultent des choix suivants :
-initialisation des variables d'état de la première copie et de leurs dérivées partielles;
-technique d'évaluation du gradient (calcul direct ou rétro propagation). Les diverses possibilités sont résumées dans les tableaux 1 et 2.

Tableau III .1 : *classification des algorithmes à partir des initialisations du système d'argument $m \in [n-N_c+2, n]$pour la modification des coefficients à l'instant n : calcul direct. Pour la ligne 1, 2 et 3, on utilise les valeurs calculées à l'instant n – 1 pour initialiser la première copie du système d'argument $m = n - N_c + 1$.*

$z_{M+1}^{1}(m)\ ,\ \dots\ ,\ z_{M+N}^{1}(m)$	$\frac{\partial z_{M+1}^{1}(m)}{\partial c_{pq}}\ ,\ \dots\ ,\ \frac{\partial z_{M+N}^{1}(m)}{\partial c_{pq}}$
1) $z_{M+1}^{2}(m-1)\ ,\ \dots\ ,\ z_{M+N}^{2}(m-1)$	$\frac{\partial z_{M+1}^{2}(m-1)}{\partial c_{pq}}\ ,\ \dots\ ,\ \frac{\partial z_{M+N}^{2}(m-1)}{\partial c_{pq}}$
2) $z_{M+1}^{2}(m-1)\ ,\ \dots\ ,\ z_{M+N}^{2}(m-1)$	$0\ ,\ \dots\ ,\ 0$
3) réponses désirées	$\frac{\partial z_{M+1}^{2}(m-1)}{\partial c_{pq}}\ ,\ \dots\ ,\ \frac{\partial z_{M+N}^{2}(m-1)}{\partial c_{pq}}$
4) réponses désirées	$0\ ,\ \dots\ ,\ 0$

Tableau III. 2 : *classification des algorithmes à partir des initialisations du système d'argument* $m \in [n-N_C+2, n]$ *pour la modification des coefficients à l'instant n : rétro propagation. Pour la ligne 1, on utilise les valeurscalculées à l'instant n – 1 pour initialiser la première copie du système d'argument* $m = n - N_C + 1$.

$z_{M+1}^{1}(m)\ ,\ \dots\ ,\ z_{M+N}^{1}(m)$	$\frac{\partial z_{M+1}^{1}(m)}{\partial c_{pq}}\ ,\ \dots\ ,\ \frac{\partial z_{M+N}^{1}(m)}{\partial c_{pq}}$
1) $z_{M+1}^{2}(m-1)\ ,\ \dots\ ,\ z_{M+N}^{2}(m-1)$	0 , … , 0 (imposé par la rétropropagation)
2) réponses désirées	0 , … , 0 (imposé par la rétropropagation)

Les tableaux présentés ci-dessus définissent autant d'algorithmes qu'il y a de lignes. Chaque algorithme est donc caractérisé (en plus des paramètres introduits précédemment comme le nombre de copies N_t, le nombre de termes N_c de la fonction de coût, le nombre d'itérations K_n du gradient, etc.) par la technique de calcul du gradient et les initialisations choisies. Bien que visant à minimiser une même fonction de coût et bien qu'utilisant un algorithme de gradient, les algorithmes donnés par les lignes des tableaux précédents sont différents.

Parmi les choix possibles d'initialisation, les lignes 1 et 4 du tableau 1 et la ligne 2 du tableau 2 sont naturelles : l'initialisation des dérivées partielles des états correspond à l'initialisation des états. Parmi ces choix naturels, la ligne 4 du tableau 1 et la ligne 2 du tableau 2 supposent que l'on connaît des réponses désirées pour toutes les variables d'état du réseau ; autrement dit, elles supposent que toutes les variables d'état sont les valeurs des sorties *y* précédentes du système. C'est le cas pour le modèle NARMAX, puisque les états sont des sorties à des instants précédents. Dans le cas plus général où l'on ne connaît pas de réponses désirées pour tous les états, seul le choix de la ligne 1 du tableau 1 est à la fois naturel et réalisable.

Des algorithmes "hybrides" utilisant les initialisations des lignes 2 et 3 du tableau 1 et celles de la ligne 1du tableau 2 sont envisageables.

Le choix du nombre de copies N_t n'est pas motivé par les mêmes considérations pour tous les algorithmes. Par exemple, pour l'algorithme de la ligne 1 du tableau 1, l'utilisation d'un grand nombre de copies n'a d'intérêt que lorsque les coefficients du système sont susceptibles d'être fortement modifiés d'une itération à l'autre (non stationnarité des signaux ou instabilité locale de l'algorithme) : à l'instant n, les coefficients $c_{pq}(n)$ sont tellement différents de ce qu'ils étaient à l'instant $n - 1$, que, pour en tenir compte dans le calcul des erreurs $e(n)$, $e(n - 1)$, … , $e(m)$, … qui vont servir à calculer $c_{pq}(n + 1)$, on a intérêt à utiliser plusieurs copies. Dans un cas stable ou stationnaire, le nombre de copies n'est pas important à partir du moment où le caractère récursif du système est conservé par le type d'initialisation. En revanche, dans le cas des algorithmes des lignes 4 du tableau 1 et 2 du tableau 2, les copies permettent de prendre en considération le caractère récursif du filtre.

La rétro propagation suppose implicitement que les dérivées partielles des entrées d'état par rapport aux coefficients sont nulles pour la première copie. Les algorithmes 2 du tableau 1 et 1 du tableau 2 conduisent donc aux mêmes modifications des coefficients; il en est de même pour les algorithmes 4 du tableau 1 et 2 du tableau 2. Comme d'autre part la rétro propagation nécessite un nombre de calculs moins important, c'estl'algorithme correspondant qui doit être utilisé.[14]

Nous allons montrer à présent que les algorithmes proposés dans la littérature pour l'adaptation des filtres récursifs et des réseaux de neurones bouclés sont des cas particuliers des algorithmes présentés ci-dessus.

III.5.3 Classification des algorithmes existants en filtrage adaptatif et pour l'apprentissage des réseaux deneurones.

Les algorithmes du filtrage linéaire récursif adaptatif entrent dans la classe plus générale des algorithmes définis ci-dessus. Ils correspondent tous à un calcul direct du gradient (tableau 1) et à $N_C=K_n=1$. L'algorithme "LMS étendu" correspond au cas $N_t=1$ et à la ligne 2. L'algorithme "RPE" (approche par erreur de sortie) coïncide avec $N_t=1$ et la ligne 1. Quant à l'algorithme avec erreur a posteriori, il est aussi associé à la ligne 1 mais avec $N_t=2$.[14]

Les algorithmes qui sont apparus récemment pour l'apprentissage supervisé de réseaux bouclés (utilisés comme filtres) entrent également dans la classification précédente. Ils correspondent tous à $N_C=1$. On peut citer l'algorithme "Tacher Forcing"(correspondant à une approche de type Equation Errer ou Série-Parallèle) qui correspond à la ligne 2 du tableau 2 pour $N_t=1$, le "Real Time Récurrent Learning Algorithme" (correspondant à une approche de type Output Errer ou Parallèle) qui correspond à la ligne 1 du tableau 1 avec $N_t=1$ et le "Truncated Backpropagation Through Time" correspondant à laligne 1 du tableau 2 avec $N_t>1$.[14]

III . 6 Concusion

Nous avons établi un cadre conceptuel général pour les algorithmes adaptatifs permettant l'apprentissage de réseaux de neurones formels non bouclés (utilisés comme filtres transverses non linéaires) ou bouclés (utilisés comme filtres récursifs non linéaires). Ces algorithmes sont fondés sur des techniques d'évaluation du gradient d'une fonction de coût. Les algorithmes classiques en filtrage adaptatif, ainsi que les algorithmes adaptatifs proposés par d'autres auteurs pour les réseaux de neurones, entrent dans ce cadre général ; de surcroît, cette approche nous permet de définir toute une famille d'algorithmes nouveaux, qui sont susceptibles d'applications dans le domaine du filtrage adaptatif non linéaire. Enfin, le caractère modulaire des algorithmes facilite les réalisations logicielles et suggère des implantations

Chapitre IV : Application à la suppression de bruit

IV.1 Introduction

Les réseaux ADALINE (neurones linéaires adaptatifs) abordés dans cette rubrique sont similaires au perceptron, mais leur fonction de transfert est linéaire plutôt que contraignante. Cela permet à leurs sorties de prendre n'importe quelle valeur, alors que la sortie du perceptron est limitée à 0 ou 1. L'ADALINE et le perceptron ne peuvent résoudre que des problèmes linéairement séparables. Cependant, ici la règle d'apprentissage LMS (moindres carrés moyens), qui est beaucoup plus puissante que la règle d'apprentissage du perceptron, est utilisée. La règle d'apprentissage LMS, ou Widrow-Hoff, minimise l'erreur quadratique moyenne et déplace ainsi les limites de décision aussi loin que possible des modèles d'apprentissage. Dans cette section, vous concevez un système linéaire adaptatif qui répond aux changements de son environnement pendant son fonctionnement. Les réseaux linéaires qui sont ajustés à chaque pas de temps en fonction de nouveaux vecteurs d'entrée et cible peuvent trouver des poids et des biais qui minimisent l'erreur quadratique du réseau pour les vecteurs d'entrée et cible récents. Les réseaux de ce type sont souvent utilisés dans les systèmes d'annulation d'erreur, de traitement du signal et de contrôle.

Dans ce chapitre explicite l'application du filtrage adaptatif basé sur les réseaux de neurones comme outil d'adaptation à la suppression du bruit d'un signal utile.

IV.2Généralité sur Filtres de réseau de neurones adaptatifs

IV.2.1 Fonctions adaptatives

Cette section présente la fonction adapt, qui modifie progressivement les pondérations et les biais d'un réseau au cours de l'apprentissage.

IV.2.2 Modèle de neurone linéaire

Un neurone linéaire avec des entrées R est illustré ci-dessous.

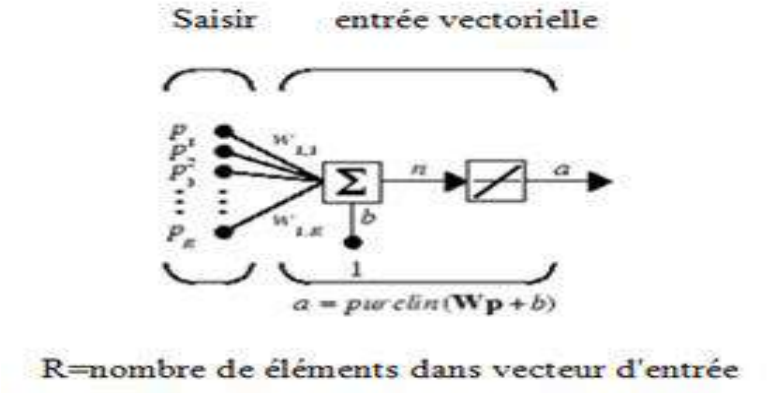


Figure IV.1 *Modèle de neurone linéaire*

Ce réseau a la même structure de base que le perceptron. La seule différence est que le neurone linéaire utilise une fonction de transfert linéaire, nommée purelin.

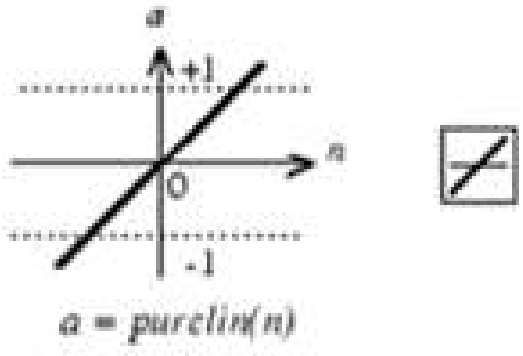


Figure IV.2 *Fonction de transfert linéaire*

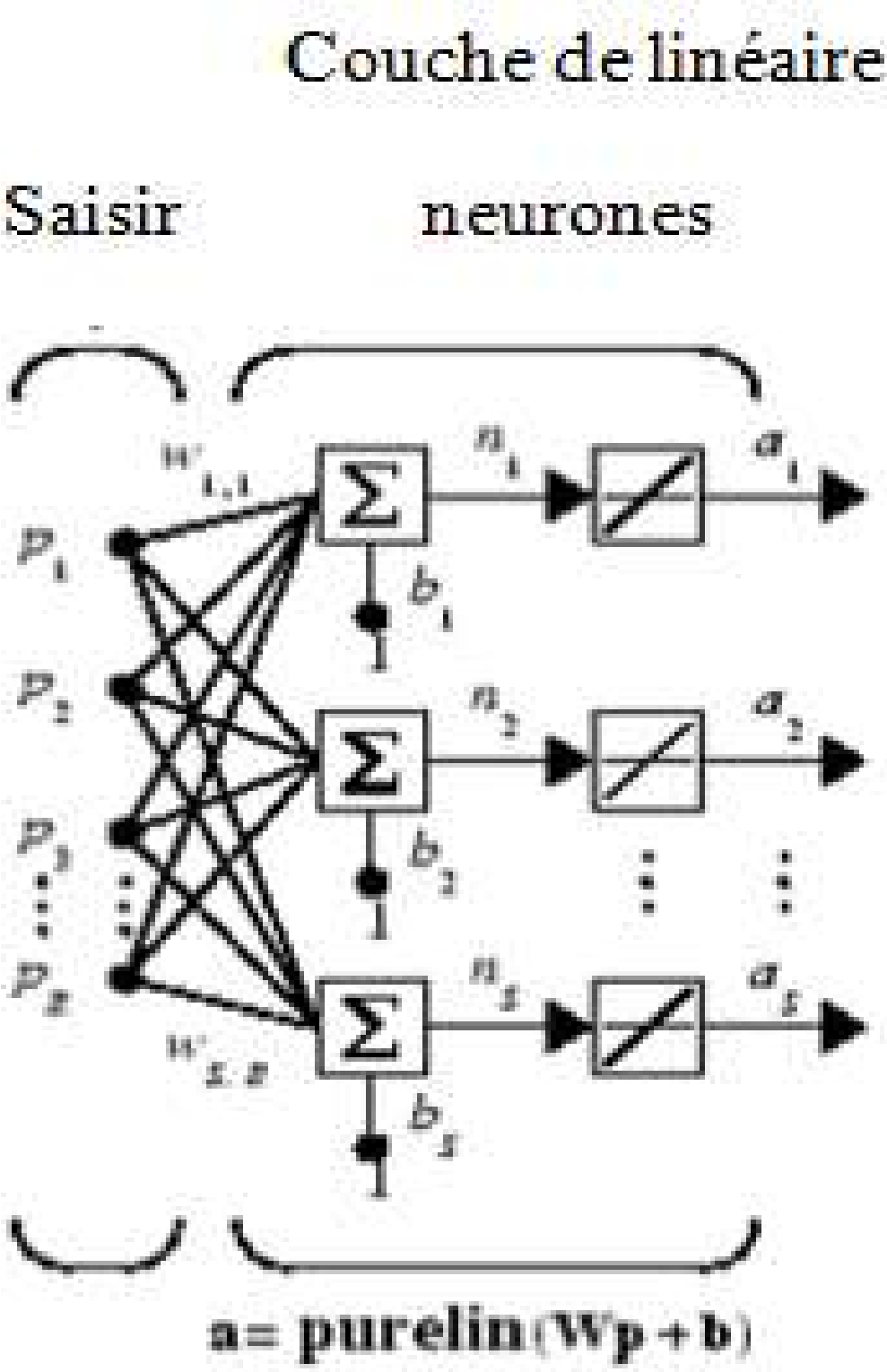
La fonction de transfert linéaire calcule la sortie du neurone en renvoyant simplement la valeur qui lui est transmise.

$\alpha = \text{purelin}(n) = \text{purelin}(Wp + b) = Wp + b$

Ce neurone peut être formé pour apprendre une fonction ane de ses entrées, ou pour trouver une approximation linéaire d'une fonction non linéaire. Un réseau linéaire ne peut bien sûr pas être conçu pour effectuer un calcul non linéaire

IV.2.3 Architecture de réseau linéaire adaptatif

Le réseau ADALINE illustré ci-dessous comporte une couche de neurones S connectés aux entrées R via une matrice de poids W



IV.3 *Réseaux linéaire adaptatif*

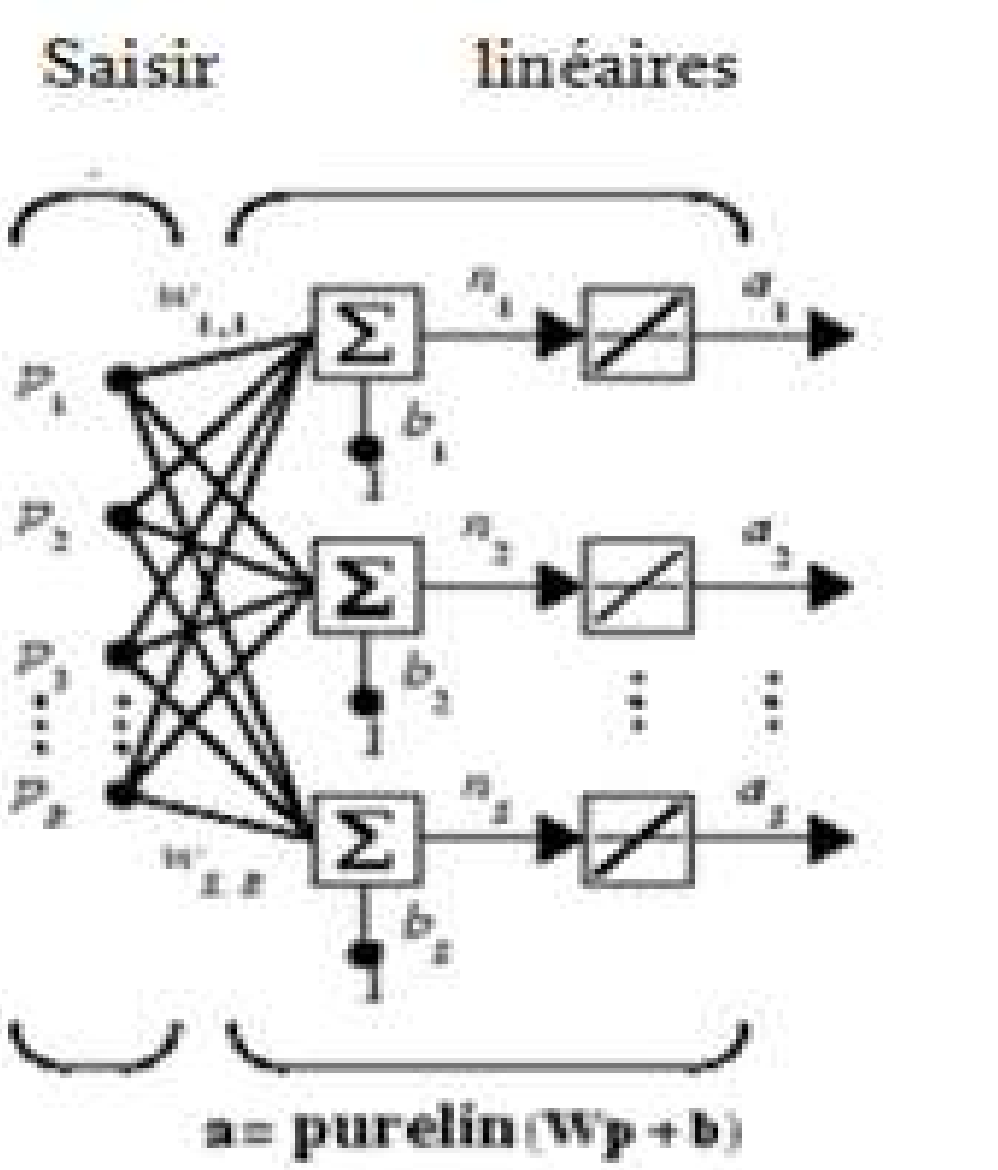


Figure IV.4 *Couche de neurones*

Ce réseau est parfois appelé MADALINE pour plusieurs ADALINE. Notez que la figure de droite dénit un vecteur de sortie de longueur S a. La règle de Widrow-Hoff ne peut former que des réseaux linéaires à une seule couche. Ce n'est cependant pas vraiment un inconvénient, car les réseaux linéaires monocouches sont tout aussi performants que les réseaux linéaires multicouches. Pour chaque réseau linéaire multicouche, il existe un réseau linéaire monocouche équivalent.

ADALINE unique (couche linéaire) .

Considérez un ADALINE unique avec deux entrées. La figure suivante montre le schéma de ce réseau.

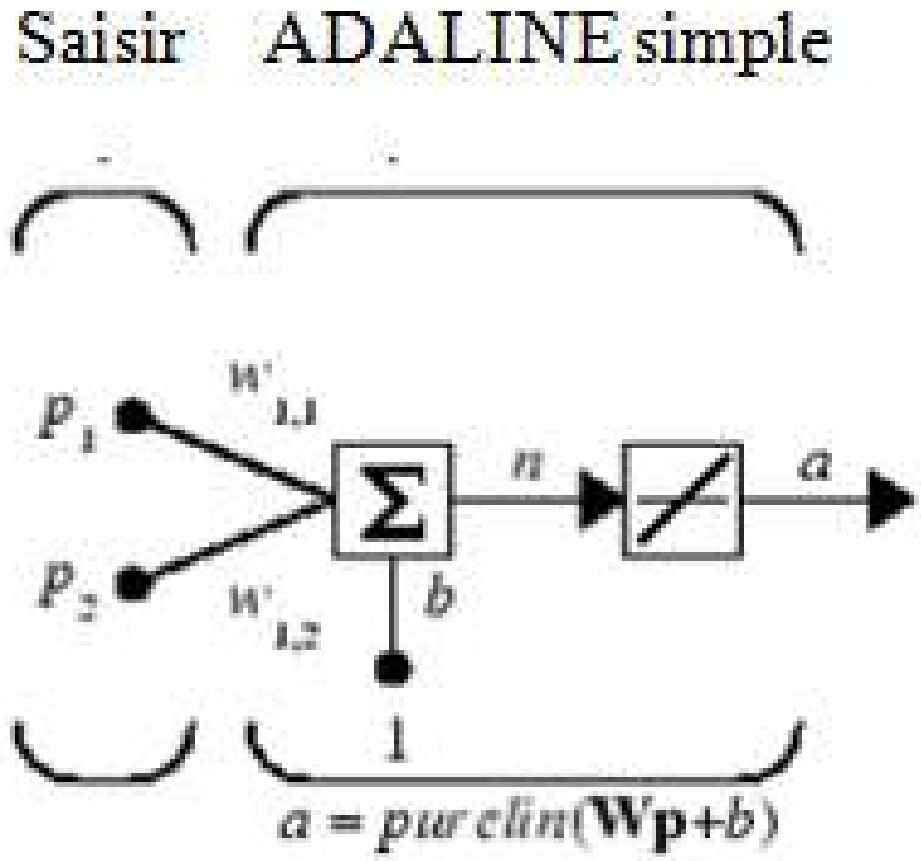
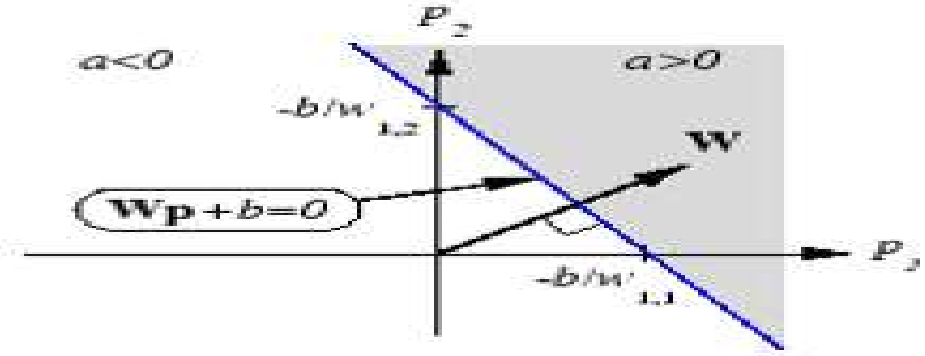


Figure IV.5 *ADALINE unique*

La matrice de poids W dans ce cas n'a qu'une seule ligne. La sortie réseau est α = purelin (n) = purelin(Wp + b) = Wp + b. Comme le perceptron, l'ADALINE a une frontière de décision qui est déterminée par les vecteurs d'entrée pour lesquels l'entrée nette n est nulle. Pour n = 0, l'équation Wp + b = 0 définit une telle limite de décision, comme indiqué ci-dessous :



Les vecteurs d'entrée dans la zone grise supérieure droite conduisent à une sortie supérieure à 0. Les vecteurs d'entrée dans la zone blanche inférieure gauche conduisent à une sortie inférieure à 0. Ainsi, l'ADALINE peut être utilisée pour classer les objets en deux catégories. Cependant, ADALINE peut classer les objets de cette manière uniquement lorsque les objets sont linéairement séparables. Ainsi, ADALINE a la même limitation que le perceptron. Vous pouvez créer un réseau similaire à celui affiché à l'aide de cette commande :

```
net = linearlayer;

net = configure(net,[0;0],[0]);
```

Les tailles des deux arguments à congurer indiquent que la couche doit avoir deux entrées et une sortie. Normalement, l'entraînement fait cette configuration pour vous, mais cela nous permet d'inspecter les poids avant l'entraînement. Les pondérations et les biais du réseau sont définis sur zéro, par défaut. Vous pouvez voir les valeurs actuelles à l'aide des commandes :

```
W = net.IW{1,1}
```

Et

```
b = net.b{1}
```

Vous pouvez également attribuer des valeurs arbitraires aux pondérations et au biais, telles que 2 et 3 pour les pondérations et -4 pour le biais :

```
net.IW{1,1} = [2 3];

net.b{1} = -4;
```

Vous pouvez simuler l'ADALINE pour un vecteur d'entrée particulier.

```
p = [5; 6];

a = sim(net,p)
```

Pour résumer, vous pouvez créer un réseau ADALINE avec linearlayer, ajuster ses éléments comme vous le souhaitez et le simuler avec sim.

IV.2.4 Erreur des moindres carrés moyens

Comme la règle d'apprentissage du perceptron, l'algorithme LMS (moindre quadratique moyen) est un exemple d'entraînement supervisé, dans lequel la règle d'apprentissage est fournie avec un ensemble d'exemples de comportement de réseau souhaité.

$\{p_1, t_1\}, \{p_2, t_2\}, \ldots \{p_Q, t_Q\}$

Ici, p est une entrée du réseau et t est la sortie cible correspondante. Lorsque chaque entrée est appliquée au réseau, la sortie du réseau est comparée à la cible. L'erreur est calculée comme la différence entre la sortie cible et la sortie réseau. Le but est de minimiser la moyenne de la somme de ces erreurs.

$\text{mse}=\frac{1}{Q}\sum_{k=1}^{Q} e(k)^2 = \frac{1}{Q}\sum_{k=1}^{Q}\big(t(k) - a(k)\big)^2$

L'algorithme LMS ajuste les poids et les biais de l'ADALINE de manière à minimiser cette erreur quadratique moyenne. Heureusement, l'indice de performance d'erreur quadratique moyenne pour le réseau ADALINE est une fonction quadratique. Ainsi, l'indice de performance aura soit un minimum global, soit un minimum faible, soit aucun minimum, vecteurs d'entrée déterminent s'il existe ou non une solution unique.

IV.2.5 Algorithme LMS (learnwh)

Les réseaux adaptatifs utiliseront l'algorithme LMS ou l'algorithme d'apprentissage Widrow-Hoff basé sur une procédure approximative de descente la plus raide. Là encore, les réseaux linéaires adaptatifs sont formés sur des exemples de comportement correct.
L'algorithme LMS, présenté ici, est décrit en détail dans Réseaux de neurones linéaires.

$\text{W}(k+1) = W(k) + 2\text{ae}(k)p^T(k)$

$\text{b}(k+1) = b(k) + 2ae(k)$

IV.2.6 Filtrage adaptatif (adapter)

Le réseau ADALINE, tout comme le perceptron, ne peut résoudre que des problèmes linéairement séparables. C'est cependant l'un des réseaux de neurones les plus largement utilisés dans les applications pratiques. Le filtrage adaptatif est l'un de ses principaux domaines d'application.

Ligne à retard taraudée

Vous avez besoin d'un nouveau composant, la ligne à retard à prises, pour utiliser pleinement le réseau ADALINE. Une telle ligne à retard est représentée sur la figure suivante. Le signal d'entrée entre par la gauche et passe par N-1 retards. La sortie de la ligne à retard à prises (TDL) est un vecteur à N dimensions, composé du signal d'entrée à l'instant courant, du signal d'entrée précédent, etc.

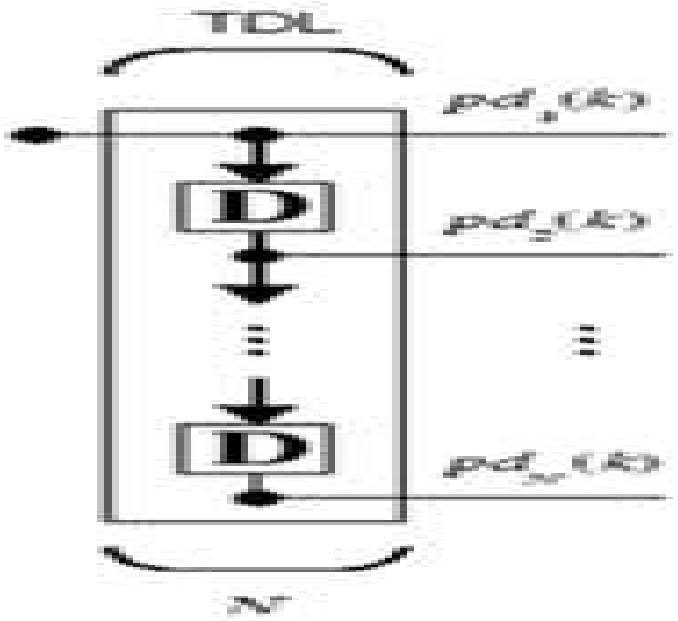


Figure IV.6 *Filtrage adaptatif*

Vous pouvez combiner une ligne à retard à prises avec un réseau ADALINE pour créer le ltre adaptatif illustré dans la figure suivante :

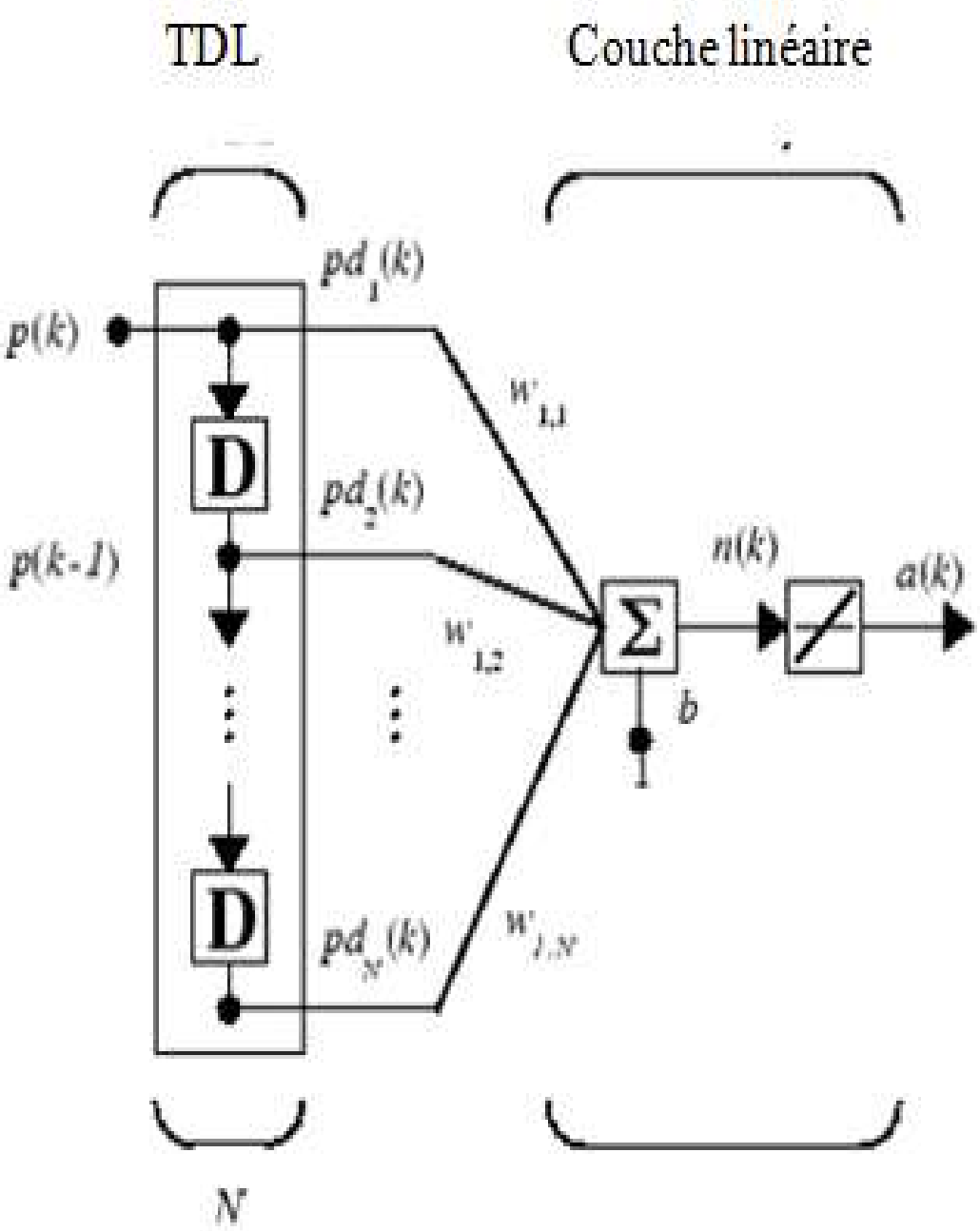


Figure IV.7 *une ligne à retard à prises avec un réseau ADALINE*

La sortie du ltre est donnée par α(k) = purelin(Wp + b)

Dans le traitement numérique du signal, ce réseau est appelé filtre à réponse impulsionnelle finie (FIR) . Jetez un œil au code utilisé pour générer et simuler un tel réseau adaptatif.

Exemple de filtre adaptatif

Tout d'abord, définissez un nouveau réseau linéaire en utilisant linearlayer.

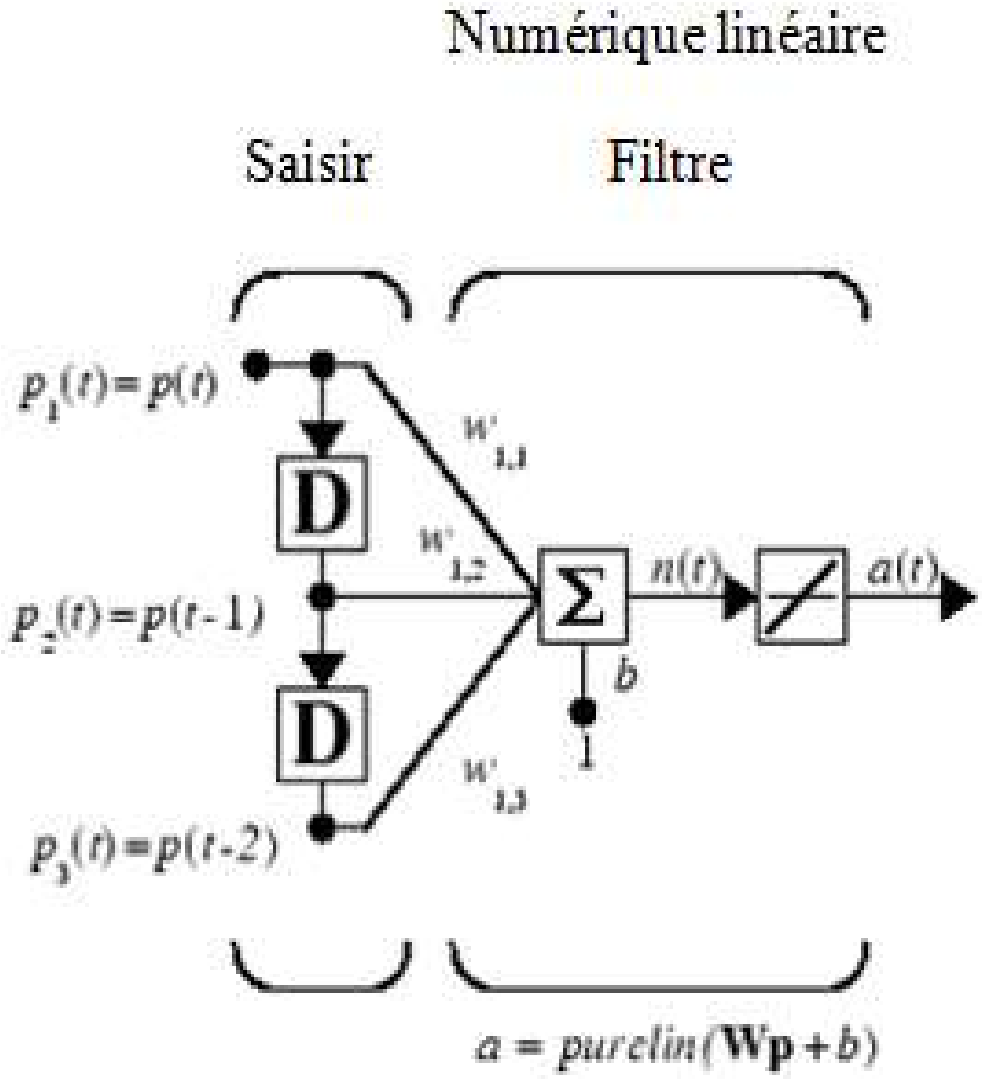


Figure IV.8 *réseau linéaire en utilisant linearlayer.*

Supposons que la couche linéaire ait un seul neurone avec une seule entrée et un retard de prise de 0, 1 et 2 retard.

```
net = linearlayer([0 1 2]);
```

```
net = configure(net,0,0);
```

Vous pouvez spécifier autant de délais que vous le souhaitez et omettre certaines valeurs si vous le souhaitez. Ils doivent être dans l'ordre croissant. Vous pouvez donner les différents poids et les valeurs de biais avec :

```
net.IW{1,1} = [7 8 9];
```

```
net.b{1} = [0];
```

Enfin, définissez les valeurs initiales des sorties des retards comme

```
pi = {1 2};
```

Ceux-ci sont ordonnés de gauche à droite pour correspondre aux délais pris de haut en bas dans la figure. Ceci conclut la configuration du réseau. Pour configurer l'entrée, supposons que les scalaires d'entrée arrivent dans une séquence : d'abord la valeur 3, puis la valeur 4, ensuite la valeur 5 et enfin la valeur 6. Vous pouvez indiquer cette séquence en définissant les valeurs comme éléments d'un tableau de cellules entre accolades.

```
p = {3 4 5 6};
```

Maintenant, vous avez un réseau et une séquence d'entrées. Simulez le réseau pour voir quelle est sa sortie en fonction du temps.

```
[a,pf] = sim(net,p,pi)
```

Cette simulation donne une séquence de sortie

```
a =
  [46]    [70]    [94]    [118]
```

Et les valeurs finales pour les sorties de retard de

```
pf =
  [5]    [6]
```

L'exemple est si simple que vous pouvez le vérifier sans calculatrice pour vous assurer que vous comprenez les entrées, les valeurs initiales des retards, etc. Le réseau que vous venez de

définir peut être entraîné avec la fonction adapt pour produire une séquence de sortie particulière. Supposons, par exemple, que vous souhaitiez que le réseau produise la séquence de valeurs 10, 20, 30, 40.

```
t = {10 20 30 40};
```

Vous pouvez entraîner le réseau défini à le faire, en partant des conditions de délai initiales utilisées ci-dessus. Laissez le réseau s'adapter pendant 10 passages sur les données.

```
For
 i = 1:10
      [net,y,E,pf,af] = adapt(net,p,t,pi); end
```

Ce code renvoie les poids finaux, le biais et la séquence de sortie indiqués ici

```
wts = net.IW{1,1}
 wts =
       0.5059    3.1053    5.7046
 bias = net.b{1}
 bias =   -1.5993
 y =
       [11.8558]    [20.7735]    [29.6679]    [39.0036]
```

Vrai semblablement, si vous avez exécuté des passes supplémentaires, la séquence de sortie aurait été encore plus proche des valeurs souhaitées de 10, 20, 30 et 40. Ainsi, les réseaux adaptatifs peuvent être spécifiés, simulés et finalement formés avec adapt. Cependant, la valeur exceptionnelle des réseaux adaptatifs réside dans leur utilisation pour remplir une fonction particulière, telle que la prédiction ou la suppression du bruit.

Exemple de prédiction

Supposons que vous vouliez utiliser un ltre adaptatif pour prédire la prochaine valeur d'un processus aléatoire stationnaire, p(t). Vous pouvez utiliser le réseau illustré dans la figure suivante pour faire cette prédiction.

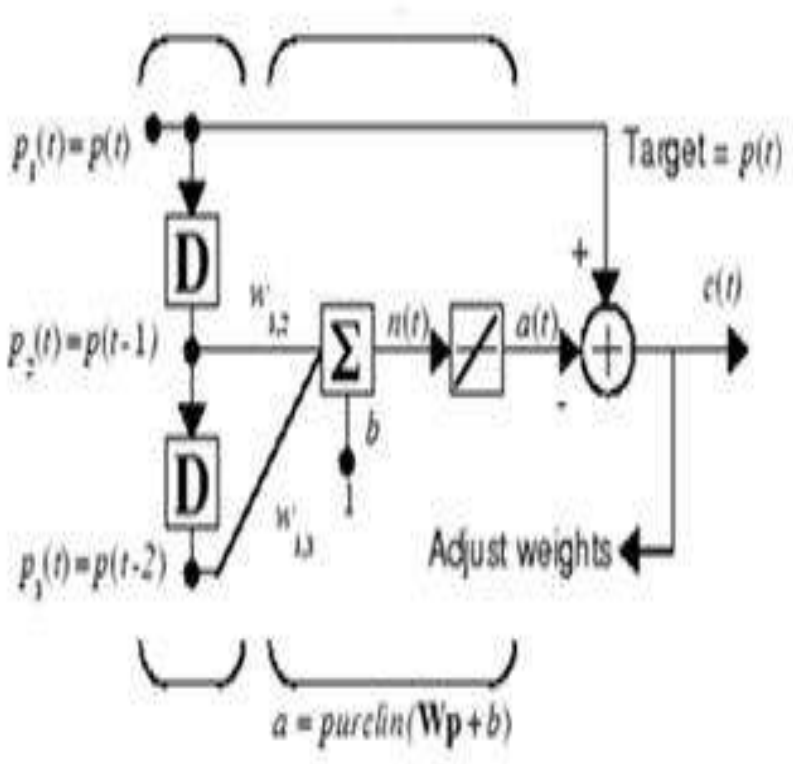


Figure IV.9 *Filtre prédictif*

Le signal à prédire, p(t), entre par la gauche dans une ligne à retard à prises. Les deux valeurs précédentes de p(t) sont disponibles comme sorties de la ligne à retard à prises. Le réseau utilise adapt pour modifier les poids à chaque pas de temps afin de minimiser l'erreur e(t) à l'extrême droite. Si cette erreur est 0, la sortie du réseau a(t) est exactement égale à p(t), et le réseau a fait sa prédiction correctement.

Étant donné la fonction d'autocorrélation du processus aléatoire stationnaire p(t), vous pouvez calculer la surface d'erreur, le taux d'apprentissage maximal et les valeurs optimales des poids. Généralement, bien sûr, vous ne disposez pas d'informations détaillées sur le processus aléatoire, de sorte que ces calculs ne peuvent pas être effectués. Ce manque n'a pas d'importance pour le réseau. Après son initialisation et son fonctionnement, le réseau s'adapte à chaque pas de temps pour minimiser l'erreur et en un temps relativement court est capable de prédire l'entrée p(t).

Le chapitre 10 de présente ce problème, passe en revue l'analyse et montre la trajectoire du poids pendant l'entraînement. Le réseau trouve par lui-même les pondérations optimales sans aucune discuté.

Vous pouvez également essayer l'exemple nnd10nc pour voir un exemple de programme de suppression adaptative du bruit en action. Cet exemple vous permet de choisir un taux d'apprentissage et un élan, et montre la trajectoire d'apprentissage, ainsi que les signaux d'origine et d'annulation en fonction du temps.

IV .3 la prédiction [16]

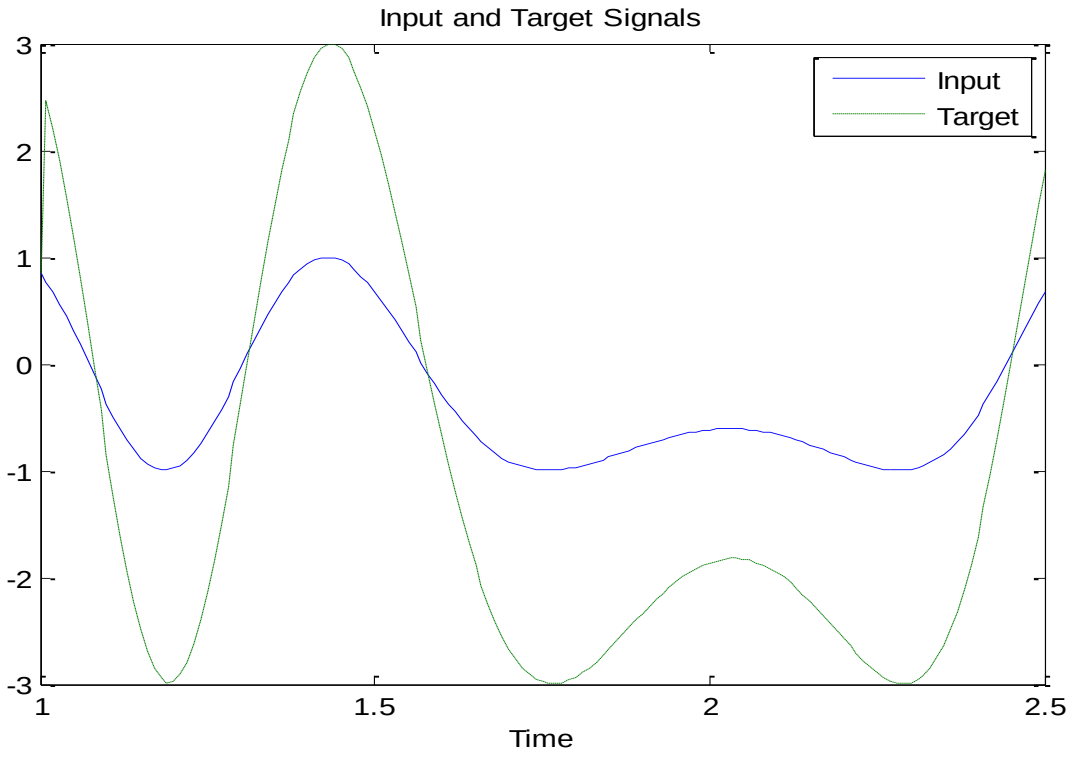
Essayez cet exemple Copier la commande Copier le code Un neurone linéaire est autorisé à s'adapter de sorte qu'à partir d'un signal, il puisse prédire un deuxième signal.

TIME définit les pas de temps de cette simulation. P définit un signal sur ces pas de temps. T est un signal dérivé de P en le décalant vers la gauche, en le multipliant par 2 et en l'ajoutant à lui-même.

```
time = 1:0.01:2.5;
X = sin(sin(time).*time*10);
P = con2seq(X);
T = con2seq(2*[0 X(1:(end-1))] + X);
```

Voici comment les deux signaux sont tracés :

```
plot(time,cat(2,P{:}),time,cat(2,T{:}),'--')
title('Input and Target Signals')
xlabel('Time')
legend({'Input','Target'})
```

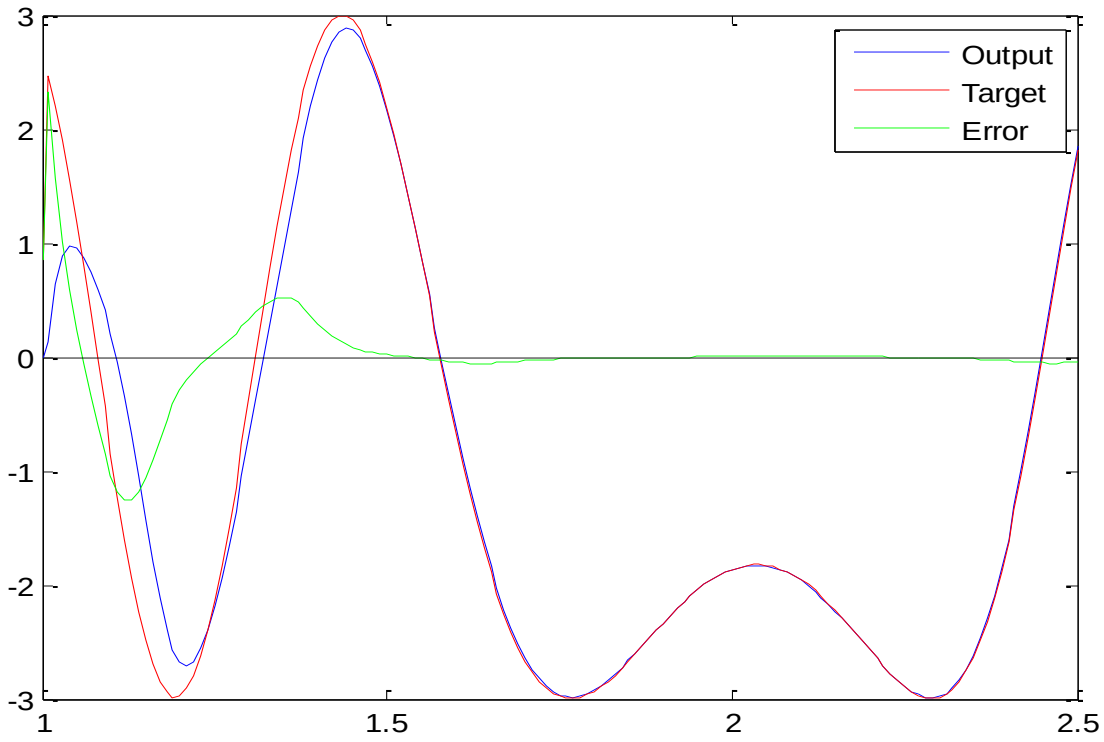


Le réseau linéaire doit avoir tapé le retard afin d'apprendre la corrélation décalée dans le temps entre P et T. NEWLIN crée une couche linéaire. [-3 3] est la plage d'entrée attendue. Le deuxième argument est le nombre de neurones dans la couche. [0 1] spécifie une entrée sans retard et une entrée avec un retard de un. Le dernier argument est le taux d'apprentissage.

```
net = newlin([-3 3],1,[0 1],0.1);
```

ADAPT simule des réseaux adaptatifs. Il prend un réseau, un signal et un signal cible, et filtre le signal de manière adaptative. Tracez la sortie Y en bleu, la cible T en rouge et l'erreur E en vert. À t = 2, le réseau a appris la relation entre l'entrée et la cible et l'erreur tombe à près de zéro

```
[net,Y,E,Pf]=adapt(net,P,T);
plot(time,cat(2,Y{:}),'b', ...
   time,cat(2,T{:}),'r', ...
   time,cat(2,E{:}),'g',[1 2.5],[0 0],'k')
legend({'Output','Target','Error'})
```



IV.4 Suppression d'une interférence

L'annulation de bruit comprend l'ensemble des techniques qui permettent d'annuler le bruit contenu dans un signal. Le signal peut représenter une image ou un son. Le bruit étant un composant indésirable, il est nécessaire de l'éliminer le plus possible pour recueillir le signal qui

nous intéresse. Les méthodes d'annulation de bruits touchent au domaine du génie électrique, plus particulièrement celui des télécommunications [15].

IV.4.1 Domaines d'application de l'annulation de bruit [15]

Ils existent de nombreuses applications à l'annulation de bruit. En voici quelques unes :
Domaine de l'aviation : l'annulation permet aux pilotes d'avion de chasse de communiquer entre eux, malgré le bruit assourdissant qui règne dans le cockpit.

Domaine de l'automobile : les pilotes de course doivent pouvoir communiquer clairement avec les membres de son équipe, sans que le bruit du moteur interférer avec le signal utile.

Domaine des téléphones portables : kit mains libres sans fils accroché permettent de communiquer clairement, bien que le microphone soit au niveau de l'oreille.

Médical: lors d'un électrocardiogramme, on obtient un signal qui contient deux informations: le battement du cœur de la mère et celui du fœtus. L'annulation de bruit permet d'annuler le battement de la mère dans le signal pour permettre l'analyse du battement du bébé.

IV.4.2 Méthodes de l'annulation de bruit

La méthode utilisée pour annuler un bruit va dépendre du contexte :

- ✓ Type de signal (stationnaire, non stationnaire).
- ✓ Type de bruit.
- ✓ Le rapport signal sur bruit (RSB).

La méthode la plus connue et la plus ancienne est l'annulation de bruit adaptive (ANC en anglais pour 'Adaptive Noise Cancellions'), c'est à dire qu'on utilise le filtrage adaptatif pour annuler le bruit [17].

IV.4.3 Principe de la méthode ANC.

Le principe est le suivant : on a un système à deux entrées. Une entrée correspond au signal qui nous intéresse d(n), l'autre entrée correspond à l'entrée secondaire, qui contient uniquement le bruit b(n). Le but du système est de retrouver en sortie un signal sans bruit, ou atténué de bruit : x(n). [17]

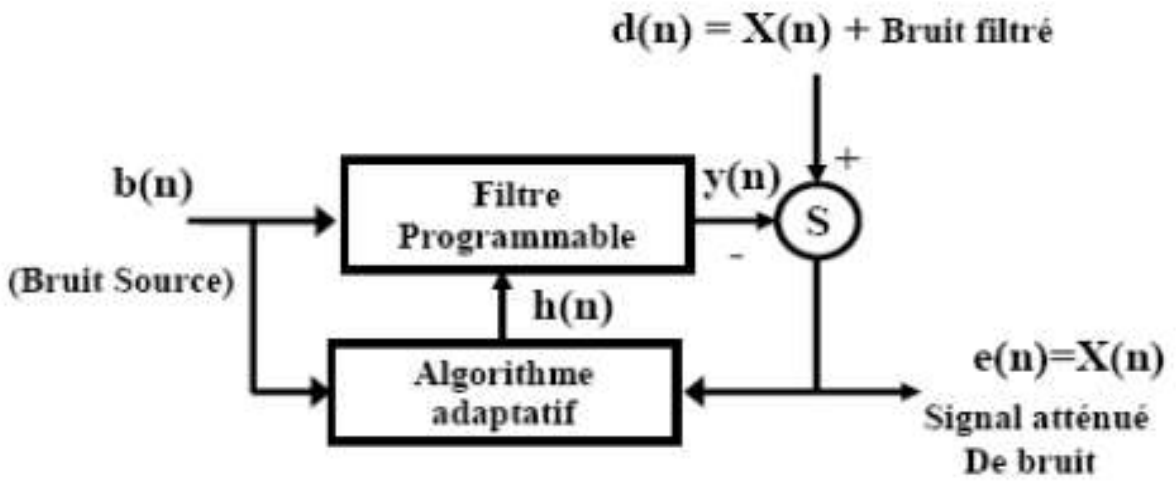


Figure IV. 10 : *Principe de la méthode « ANC »*

IV.4.4 Explication de la méthode.

Avec un microphone, vous enregistrez une conversation signal (d(t)=x(t)+bf(t)) dans un lieu particulièrement bruyant (salle de machines, intérieur d'un véhicule motorisé, salle mal isolée phoniquement de l'extérieur ou d'une salle voisine où se déroule une autre activité, lieu public extérieur fréquenté). La qualité de l'enregistrement sera alors fortement affectée par la présence du parasite. Cependant, nous pouvons placer un second micro éloigné du premier, qui n'enregistre que le parasite : c'est le signal b(t). Bien sûr, qu'il est corrélé au bruit accompagnant le signal bruité. BF(t) mais n'est pas b(t) lui-même donc, il ne peut être envisagé d'extraire le signal en effectuant une simple soustraction x (t)=d(t)-b(t). Donc l'opération générale peut se schématiser comme suit :

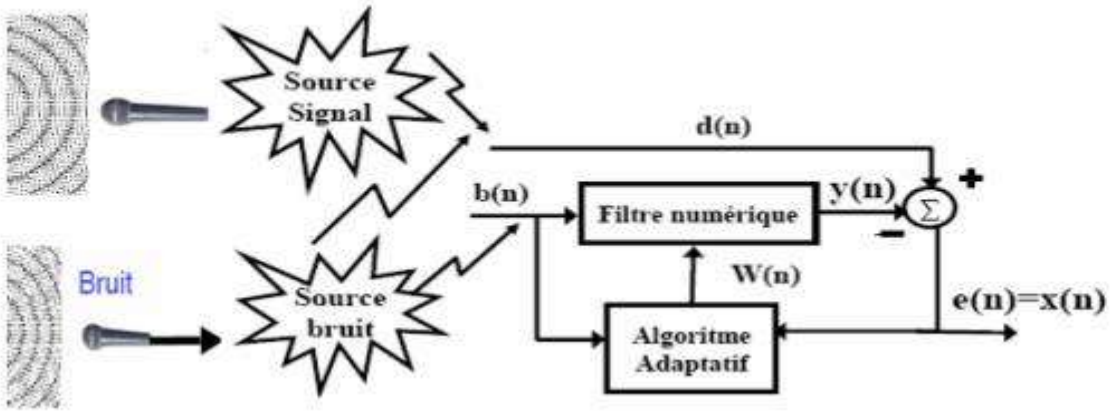


Figure IV.11 : *Montage de filtre adaptatif pour l'annulation de bruit.*

Notez que: Dans la suppression du bruit, par les filtres adaptatifs nous permettent de supprimer le bruit d'un signal en temps réel. Ici, le signal mesuré d(t), l'un pour nettoyer, qui est l'entrée du filtre adaptatif, Le bruit b(n) est le signal de référence du filtre adaptatif, qui représente le signal corrélée avec le bruit à supprimer du signal désiré d(n).Le filtre adaptatif ajuste et adapte ses coefficients pour réduire la valeur de la différence entre y(n) et d (n) ; supprime le bruit et entraînant un signal propre en e(n).

Notez que dans cette application, le signal d'erreur en fait converge vers le signal de données utile, plutôt que de la convergence à zéro.

IV.5 Résultats de simulation

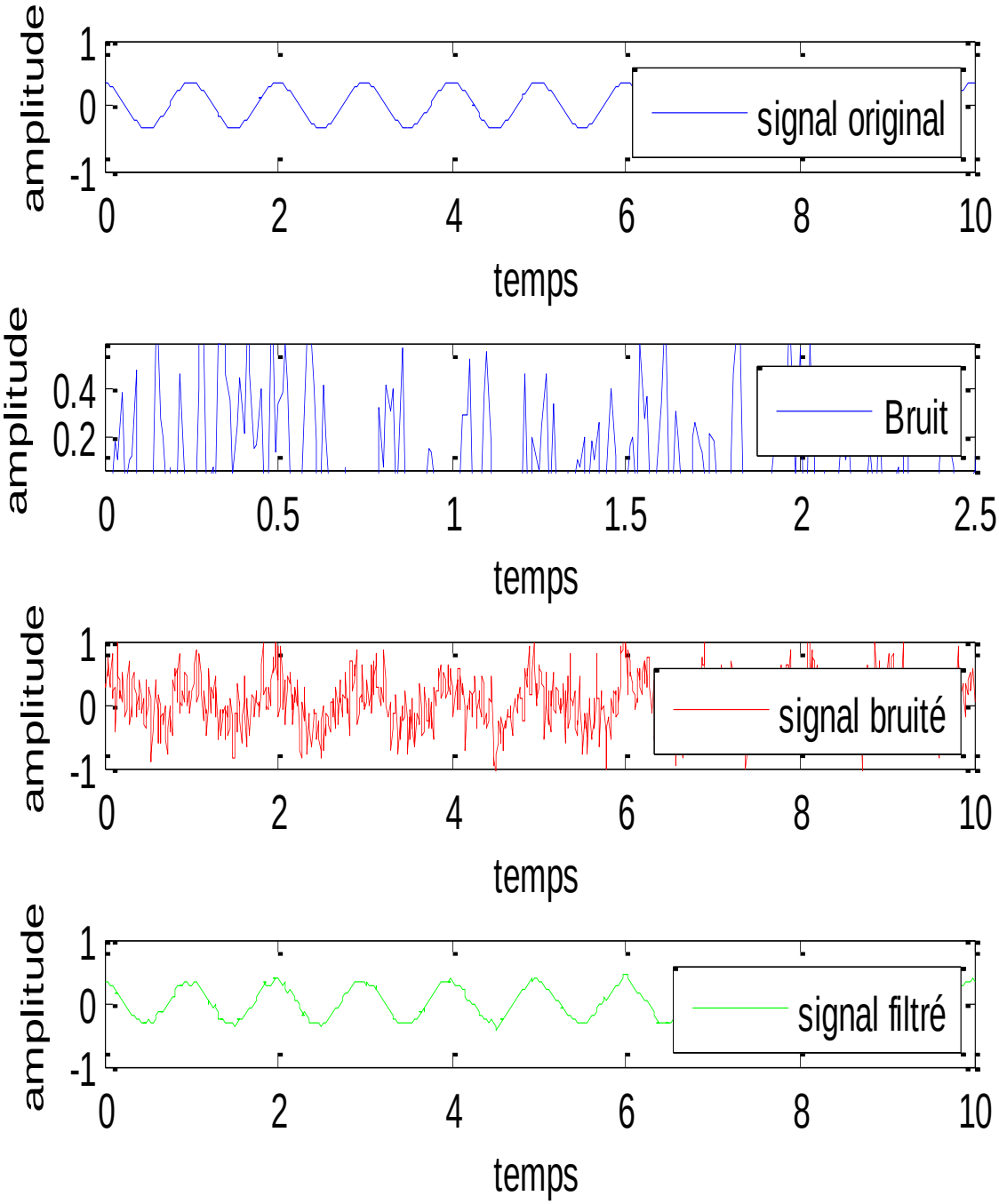


Figure IV .12 *Réduction du bruit dans un signal sinus à fréquence variable*

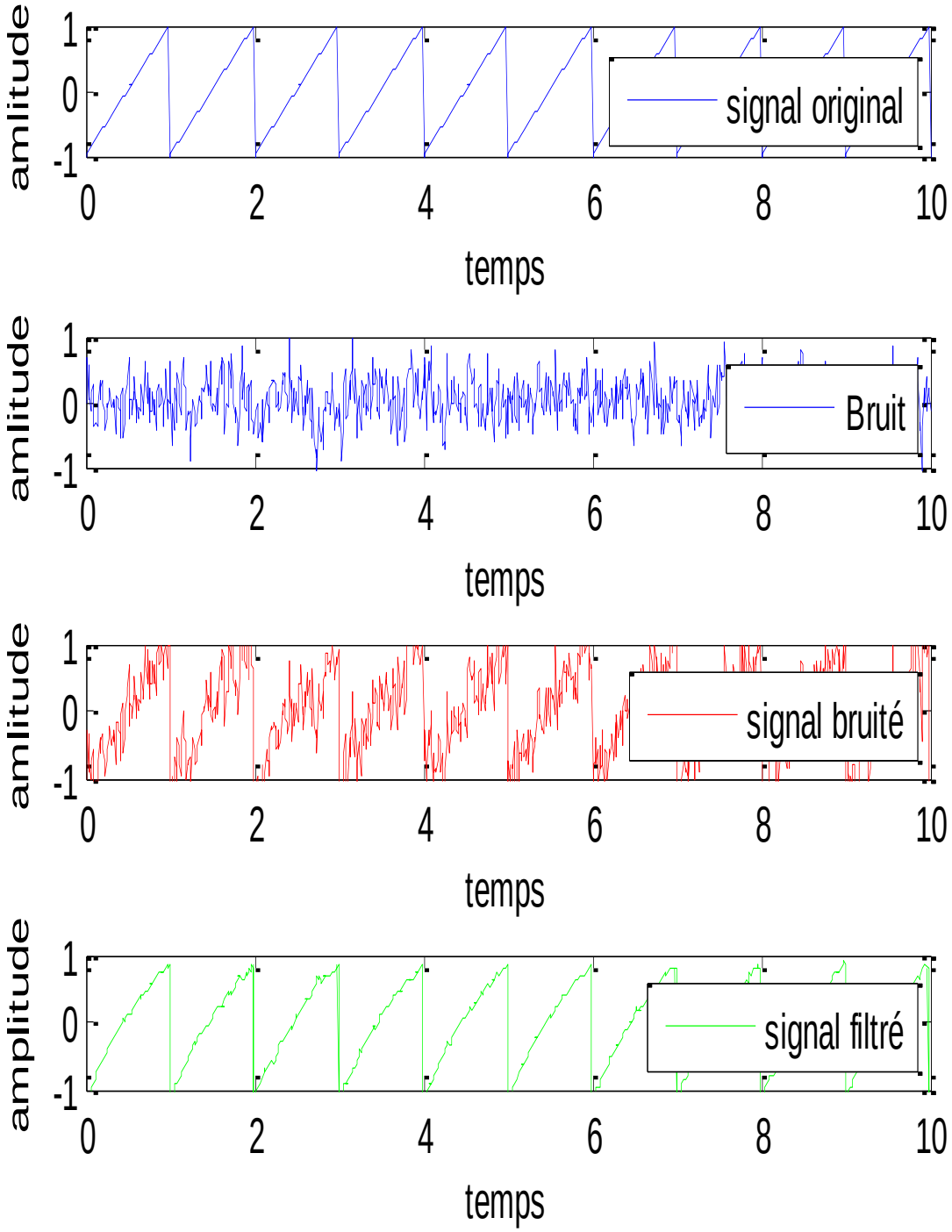


Figure IV .13 *Réduction du bruit dans un signal a dents scie (sawtooth)*

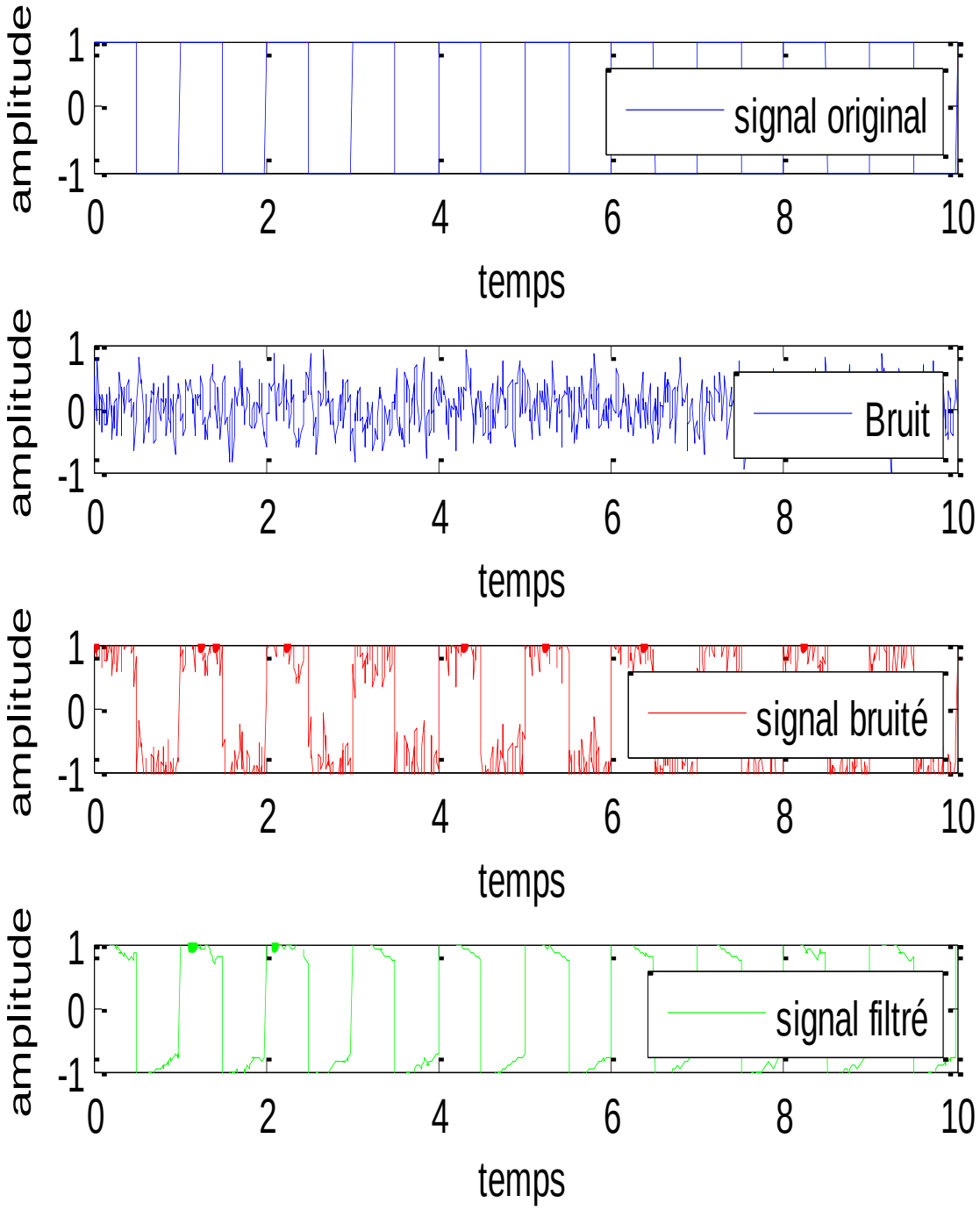


Figure IV.14 *Réduction du bruit dans un signal squar*

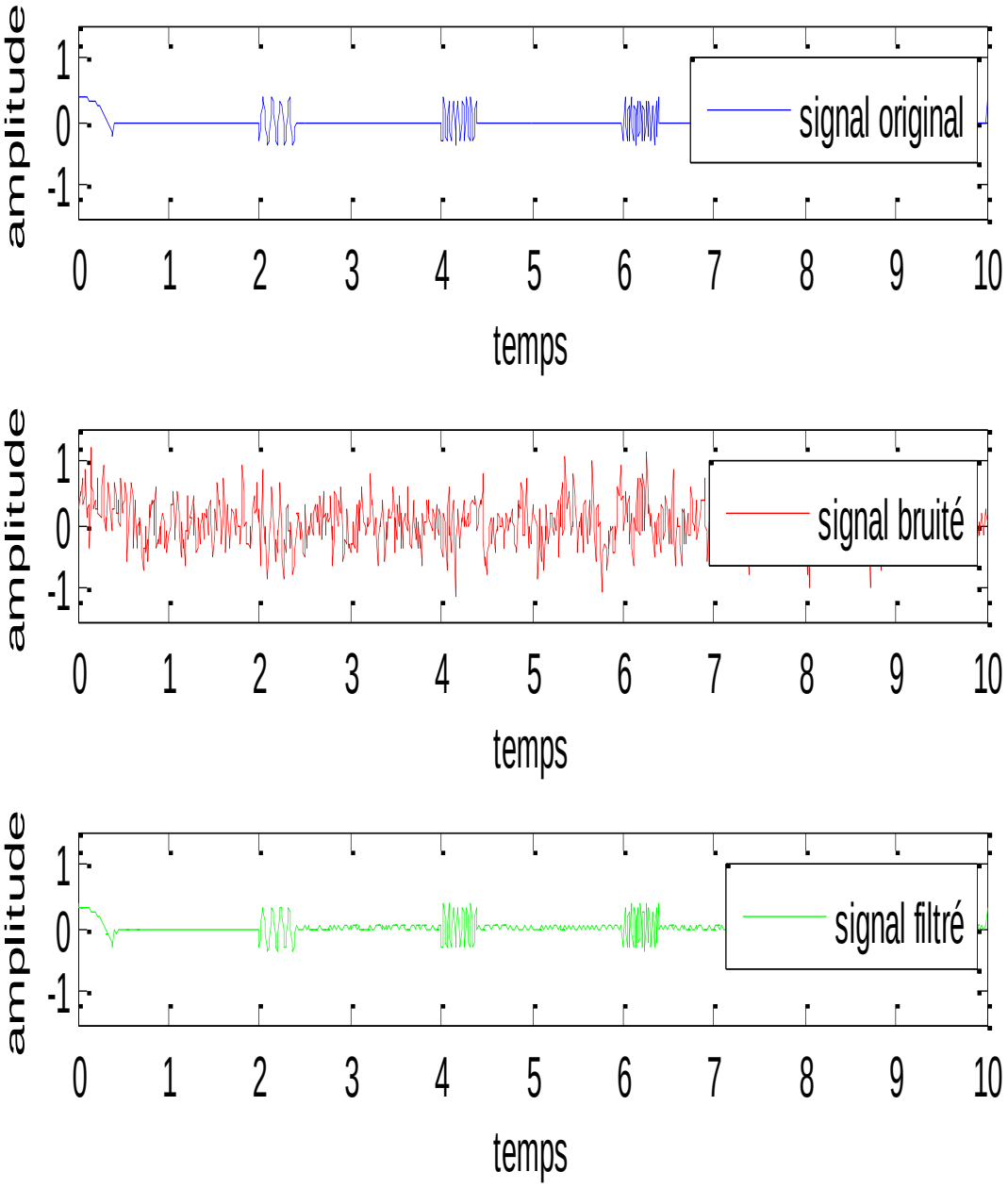


Figure IV.15 *Réduction du bruit dans un signal(le signal variable*signal square)*

IV.6. Discussion

Pour réduire le bruit du signal, le filtre est souvent utilisé dans la chaîne de traitement du signal, de la rétroaction initiale au résultat final, la première étape étant souvent l'élimination des signaux parasites. Ceci est possible lorsque ce dernier a des propriétés différentes de celles du signal.

IV .7. Conclusion

Conclu dans l'application pour supprimer le bruit, Les résultats de la simulation pour la réduction du bruit de différents types de signaux, sinusoïdaux, carrés et en dents de scie immergés dans le bruit, d'abord à fréquence constante puis à fréquence variable, ont montré l'efficacité et la justesse d'exploiter le réseau comme filtre adaptatif.

Conclusion Générale

Conclusion générale

Ce projet, nous a permis d'approfondir notre connaissance dans le domaine du traitement de signal et plus précisément dans le domaine du filtrage adaptatif en utilisant de réseaux de neurones artificiels.

Nous avons établi un cadre conceptuel général pour les algorithmes adaptatifs permettant l'apprentissage de réseaux de neurones formels non bouclés (utilisés comme filtres transverses non linéaires) ou bouclés (utilisés comme filtres récursifs non linéaires). Ces algorithmes sont fondés sur des techniques d'évaluation du gradient d'une fonction de coût.

Les principales qualités des réseaux de neurones sont leur capacité d'adaptabilité et d'auto-organisation et la possibilité de résoudre des problèmes non-linéaires avec une bonne approximation. Ils ont une bonne immunité aux bruits et se prêtent bien à une implantation parallèle. La rapidité d'exécution est une qualité importante et elle justifie souvent à elle seule le choix d'implanter un réseau de neurones. Ces qualités ont permis de réaliser avec succès, plusieurs applications : classification, filtrage, compression de données, contrôleur, etc.…

Ce mémoire est consacré à la suppression de bruit adaptative des signaux, en utilisant des structures à base des réseaux de neurones artificielles.

Notre choix des réseaux de neurones comme outils de filtrage adaptatif pour la suppression de bruit est une continuité d'un autre travail réalisé par autres étudiants, qui ont appliqué les algorithmes du filtrage adaptatif classique (RLS, LMS NLMS) au filtrage d'un signal radar à fréquence variable.

La similitude qui existe entre la structure d'un filtre d'adaptatif de type RIF et un réseau de neurone ADALINE, dont les poids du réseau sont les coefficients du filtre RIF, et le parallélisme de traitement dans le cas d'un RNA nous a permis de bien les exploiter comme filtre adaptatif pour la suppression du bruit d'un signal.

Les résultats de simulation pour le dé bruitage de différents types de signaux, sinus, carré, dents de scies noyées dans un bruit, premièrement à une fréquence fixe puis à une fréquence variable, ont montré l'efficacité et la justesse de fonctionnement du réseau comme un filtre adaptatif.

La précision dépend du nombre d'itérations exigées par l'utilisateur, qui a un effet direct sur la vitesse de traitement. Cette dernière même dépend des contraintes de l'application envisagée. Comme perspective notre projet peut être le sujet d'une implémentation hardware sur une plateforme programmable (FPGA ou DSP).

Bibliographie

Bibliographie

[1] FERRANE ,khaled et SADOUNI ,ibtissem .Filtrage adaptatif pour l'annulation d'écho acoustique dans les communication mains libres .46p .Mémoire de Master :Génie électrique :Alger, Faculté des sciences et des sciences application :2017.

[2] BERRABAH ,randa et MOHAMED MAAMAR ,fadia .Filtre adaptatif IPNLMS à complexité réduite basé sur la mise à jour sélective de blocs de coefficients .88p . Mémoire de Master :Réseaux & Télécommunication :Blida, Université SAAD DAHLAB.2019/2020.

[3] HAMIDIA ,Mahfoud .Rehaussement de la parole par filtrage adaptatif pour les application en télécommunications .83p . Mémoire de Magister :Communication Parlée :Alger, Université des Sciences et la Technologie Houari BOUMEDIENE.2012.

[4] CHETTIBI ,ismahane et ZENIFECHE ,bilal .Commande adaptative par Réseaux de Neurones Artificiel d' un Système non Linéaire incertain .49p.These de Master :Automatique et Informatique Industrielle :Jijel, Faculté des Sciences et Technique .2017/2018.

[5] A. zilouchain and M. Jumshide: Intelligent Control Systems Using Soft Computing Methodologies. CRC Press LLC, 2001.

[6] B. Gosselin, « Application de réseaux de neurones artificiels a la reconnaissanceautomatique de caractères manuscrits, » thèse de doctorat, Faculté Polytechniquede Mons, 1996.

[7] C. Touzet, "Les réseaux de neurones artificiels : Introduction auconnexionnisme", Ed. Scribd, San Francisco, USA, 1992.

[8] H. Boudjdir, «Commande adaptative par les approximateur universels des systèmes non linéaire», Thèse de Doctorat, Université de jijel, 2015.

[9] Cyril Voyant. Prédiction de séries temporelles de rayonnement solaire global et de production d'énergie photovoltaïque à partir de réseaux de neurones artificiels. Energie électrique. Université Pascal Paoli, 2011. Français.

[10] K. HORNIK, "Multilayer Feedforward Networks are Universal Approximators", Neural Networks, Vol.2, pp. 359-366, 1989.

[11] A. WAIBEL , T. HANAZAWA, G. HINTON, K. SHIKANO, and K. LANG, "Phoneme Recognition Using Time-Delay Neural Networks", IEEE Trans. on Acoustics, Speech, and Signal Processing 37, 328-339, 1989.

[12] L. PERSONNAZ, O. NERRAND, G. DREYFUS, "Apprentissage et Mise en Oeuvre de Réseaux de Neurones Bouclés", Journées Internationales des Sciences Informatiques, Tunis, 1990.

O. NERRAND, P. ROUSSEL-RAGOT, L. PERSONNAZ, G. DREYFUS, S. MARCOS, "Training Discrete-Time Feedback Networks for Filtering and Control", Neural Network World, vol. 1, pp. 205-212, 1991.

[13] O. NERRAND, P. ROUSSEL-RAGOT, L. PERSONNAZ, G. DREYFUS, S. MARCOS, O. MACCHI, C. VIGNAT, "Neural Network Training Schemes for Non-linear Adaptive Filtering and Modelling", Proceedings of IJCNN, Seattle, 1991.

O. NERRAND, P. ROUSSEL-RAGOT, L. PERSONNAZ, G. DREYFUS, S. MARCOS, "Neural Networks and Non Linear Adaptive Filtering: Unifying Concepts and New Algorithms", soumis pour publication.

[14] OLIVER,nerrand et GERARD,dreyfus. Réseaux de neurones pour le filtrage non linéaire adaptatif .27p .paris .Ecole d'électronique .2014.

[15] BENCHENIEF, Abderrazak. Réduction du bruit d'un signal monodimensionnel par le filtrage adaptatif récursif.55p.Memoire de fin d'étude en vue de l'obtention du diplôme d'ingénieur d'état en électronique : Faculté des sciences et de la technologie .2009.

[16] Manuel Matlab 2021**.**Neural Network Toolbox Examples.

[17] "http://www.w3.org/1999/xhtml">< Annulations de bruit Wikipédia >"

Bibliographie