N° d'ordre : ……

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE D'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE
« HOUARI BOUMADIENE »
FACULTE DES SCIENCES DE LA TERRE, DE LA GEOGRAPHIE ET DE
L'AMENAGEMENT DU TERRITOIRE (F.S.T.G.A.T)

MEMOIRE

Présenté pour l'obtention du diplôme de **MAGISTER**

En : **Sciences de la terre**

Spécialité : **Géophysique**

Par : **M. KAMEL Karim**

Sujet

RESEAUX DE NEURONES EN GEOPHYSIQUE.
APPLICATION AUX DONNEES SISMIQUES ET ELECTRIQUES.

Soutenu le 26/05 / 2005, devant le jury composé de :

Pr. IDRES Mouloud, F.S.T.G.A.T, USTHB.	Président
Pr. BAKER Hayder Aziz, F.S.T.G.A.T, USTHB.	Rapporteur
Pr. DJEDDI Mebrouk, Université de Boumerdès.	Examinateur
Dr. DJEDDI Mohamed, Maître de Conférence, F.S.T.G.A.T, USTHB.	Examinateur
M^{me} ZAOURAR Naima, Chargé de Cours, F.S.T.G.A.T, USTHB.	Examinateur

Dédicace

A Ma mère, à Mon père

A mes frères et sœurs

A tous mes enseignants.

. A tous mes amis

Je dédie ce modeste travail.

Remerciements

Je tiens à remercier vivement mon promoteur **Pr. BAKER Hayder Aziz** pour son suivi et ses orientations durant la réalisation de ce mémoire.

Aussi, Je dois remercier le président du jury **Pr. IDRES Mouloud,** les membres de jury **Pr. DJEDDI Mebrouk**, **M^{me} ZAOURAR Naima** et **Dr. DJEDDI Mohamed** pour le temps qu'ils le consacrent à la lecture de ce mémoire.

Je remercie également mes enseignants de département Géophysique. De même, Je ne n'oublies pas les aides et les conseils apportées par tous mes amis notamment **N. HELLAL, S. GACI**, **S. IDIR, S. SEMAI, B. SEMAR** et **A**. **BENZINNE.**

J'exprime ma profonde gratitude à l'ensemble des personnes qui m'ont aidé de prés ou de loin, à la réalisation de ce mémoire.

Sommaire

CHAPITRE III : INVERSION DES SONDAGES ELECTRIQUES PAR RNA

CHAPITRE IV : APPLICATIONS

RESEAUX DE NEURONES ARTIFICIELS EN GEOPHYSIQUE.
APPLICATION AUX DONNEES SISMIQUES ET ELECTRIQUES

Présenté par : KAMEL Karim

Faculté des Sciences de la Terre, de la Géographie et de l'Aménagement du Territoire (F.S.T.G.A.T)
Département de Géophysique

RESUME

Les réseaux de neurones artificiels dont leur mécanisme de fonctionnement est principalement inspiré du cerveau humain sont entrain d'envahir plusieurs domaines de la science et de la technologie.
Contrairement aux méthodes classiques qui utilisent des algorithmes spécifiques pour la résolution d'un problème donné, les réseaux de neurones artificiels utilisent un schéma d'apprentissage « *Learning* » qui consiste à élaborer des relations entre un nombre suffisamment élevé des intrants « *Input* » et des sortants « *Ouput* », permettant d'acquérir des informations importantes sur le problème en question.

La capacité d'apprentissage et d'adaptation de ces modèles leur permet d'être attractifs pour des applications en géophysique. En effet, les réseaux de neurones artificiels ont réussi à rendre certaines tâches de traitement sismique automatiques telles que : l'édition, le pointé des premières arrivées, l'analyse des vitesses et les pointés des horizons sismiques.

Dans notre travail, nous avons utilisé cette approche pour la résolution de deux problématiques, une qui peut être ramenée à un problème de reconnaissance de forme « *Pattern Recognition* », et l'autre qui est un problème d'inversion.

Dans la première application, il s'agit de pouvoir pointer un horizon sismique par le bais d'un réseau de neurone apprenti sur une forme de référence choisie de la section à laquelle cette technique est appliquée. Nous avons testé la performance de cette méthode sur des données générées synthétiquement et sur une section réelle. Par la suite, les résultats ont été comparés à ceux obtenus par une méthode classique dite la méthode des moments invariants.

Dans la seconde application, il s'agit d'inverser les courbes du sondage électrique via un réseau de neurone apprenti sur un nombre important des modèles de résistivité. Ce dernier est appliqué sur des données synthétiques et sur des données réelles. Ensuite, les résultats obtenus ont été comparés à ceux obtenus par une autre technique appelée recuit simulé « *Simulated Annealing* ».

Dans les deux applications suscitées, nous avons utilisé le logiciel Pythia spécialisé dans la conception des réseaux de neurones artificiels, tandis que nous avons utilisé le langage Fortran pour les méthodes classiques.

Thèse de Magistère **Option :** Géophysique.
Directeur de thèse : Pr. BAKER Hayder Aziz, F.S.T.G.A.T, USTHB.
Mots clés : intelligence artificielle, réseaux de neurone artificiels, rétro- propagation, sondage électrique, recuit simulé, moments invariants.

Introduction

INTRODUCTION

Les réseaux de neurones artificiels dont leur mécanisme de fonctionnement est principalement inspiré du cerveau humain sont entrain d'être utilisés dans plusieurs domaines de la science et de l'ingénierie.
Contrairement aux méthodes classiques qui utilisent des algorithmes particuliers pour la résolution d'un problème spécifique, les réseaux de neurones artificiels utilisent un schéma d'apprentissage « *Learning* » qui consiste à élaborer des relations entre un nombre suffisamment élevé des entrants « *Inputs* » et des sortants « *Ouputs* », permettant d'acquérir des informations importantes sur le problème en question.

La capacité d'apprentissage et d'adaptation de ces modèles leur permet d'être attractifs pour des applications en géophysique. En effet, les réseaux de neurones artificiels ont réussi à rendre certaines tâches de traitement sismique automatiques telles que : l'édition, le pointé des premières arrivées, l'analyse des vitesses et le pointé des horizons sismiques.

Cette technique a été également utilisée dans l'interprétation de diagraphie et dans la caractérisation des réservoirs. Une autre application très importante de RNA (réseaux de neurones artificiels) est l'inversion en géophysique.

Dans notre travail, nous avons utilisé cette approche pour la résolution de deux problématiques, une qui peut être ramenée à un problème de reconnaissance de forme « *Pattern recognition* », et l'autre qui est un problème d'inversion. Il est intéressant de signaler que le réseau de neurones est capable de résoudre des problèmes de prédiction.

Dans la première application, il s'agit de pouvoir pointer un horizon sismique par le biais d'un réseau de neurone apprenti sur une forme de référence choisie de la section à laquelle cette technique est appliquée. Nous avons testé la performance de cette méthode sur des données générées synthétiquement et sur une section réelle. Par la suite, les résultats ont été comparés à ceux obtenus par une méthode classique dite la méthode des moments invariants.

Dans la seconde application, il s'agit d'inverser les courbes du sondage électrique, via un réseau de neurone apprenti sur un nombre important des modèles de résistivité. Ce réseau est appliqué sur des données synthétiques et sur des données réelles. Ensuite, les résultats obtenus ont été comparés à ceux obtenus par une autre technique appelée recuit simulé « *Simulated Annealing* ». Enfin, nous avons tenté de combiner les deux techniques pour avoir des résultats meilleurs.

Pour cela, notre mémoire est composé de quatre chapitres. Dans le premier chapitre, nous avons présenté la théorie des réseaux de neurones artificiels. Ensuite, nous avons réservé le deuxième chapitre à une description des techniques utilisées pour le pointé des horizons sismiques. Quant au troisième chapitre, il est consacré à l'inversion des courbes de sondage électrique par les réseaux de neurones artificiels et par la technique recuit simulé « *Simulated Annealing* ».

Dans le dernier chapitre, nous avons présenté les différentes applications des réseaux de neurones au pointé des horizons sismiques et à l'inversion des courbes de sondage électrique. Afin de juger la performance de la dite technique, nous avons utilisé également d'autres techniques classiques.
Enfin, nous terminons notre mémoire par une conclusion et quelques recommandations.

Chapitre I

I.1. INTRODUCTION

Les réseaux de neurones artificiels (ou réseaux connexionnistes) sont fondés sur des modèles qui tentent d'expliquer comment les cellules du cerveau et leurs interconnexions parviennent, d'un point de vue global, à exécuter des calculs complexes.

Ces systèmes, qui stockent et retrouvent l'information de manière similaire au cerveau, sont particulièrement adaptés aux traitements en parallèle des problèmes complexes comme la reconnaissance automatique de la parole, la reconnaissance de visages ou bien la simulation de fonctions de transfert. Ils offrent donc un nouveau moyen de traitement de l'information.

Les architectures connexionnistes s'inspirent de l'organisation neuronale du cerveau humain. Dans les réseaux de neurones artificiels, de nombreux processeurs appelés cellules ou unités, capables de réaliser des calculs élémentaires, sont structurés en couches successives capables d'échanger des informations au moyen de connexions qui les relient. On dit de ces unités qu'elles miment les neurones biologiques. Grâce à ce parallélisme massif, on peut espérer pouvoir surmonter les problèmes posés par des temps d'attente importants caractéristiques à la résolution de tâches complexes par des méthodes numériques.

Nous donnerons dans ce chapitre les notions de base pour la compréhension des réseaux de neurones.

I.2. HISTORIQUE

Le champ des réseaux de neurones commence par la présentation en 1943 par W. MCCulloch et W. Pitts du neurone formel qui est une abstraction du neurone physiologique.
En 1949, D. Hebb présente dans son ouvrage une règle d'apprentissage. De nombreux modèles de réseaux aujourd'hui s'inspirent encore de la règle de Hebb.
En 1958, F. Rosenblatt développe le modèle de Perception. C'est un réseau de neurones inspiré du système visuel. Il possède deux couches de neurones : une couche de perception et une couche liée à la prise de décision. C'est le premier système artificiel qui est capable d'apprendre par expérience.
En 1969, M. Minsky et S. Papert publient une critique des propriétés du Percepton. Cela va avoir une grande incidence sur la recherche dans ce domaine. Elle va fortement diminuer jusqu'en 1972, où T. Kohonen présente ses travaux sur les mémoires associatives et propose des applications à la reconnaissance de formes.
Durant la fin des années 1970, les minis, puis les micro-ordinateurs deviennent aisément disponibles. Il devient possible de construire des simulations de système complexes. Ceci est important parce que les simulations de réseaux de neurones demandent d'importants calculs.
Les limites de l'approche traditionnelle en intelligence artificielle commencent à se faire sentir et le domaine des réseaux de neurones est maintenant prêt, à la fin des années soixante-dix, pour un regain d'intérêt. Par exemple, Hinton et Anderson publient en 1981 le premier ouvrage sur les mémoires associatives parallèles.

On attribue à Hopfield un rôle majeur dans cette résurrection. Elle est due également au groupe Mcclelland et Rumelhart dans lequel se trouve entre autres Crick, qui a obtenu avec Waltson le prix Nobel pour leurs travaux sur la structure en double hélice de l'ADN. Ils ont montré et répété que des règles d'apprentissage permettaient aux réseaux de neurones d'apprendre des fonctions que le Perceptron ne pouvait comprendre (c'est la technique de la rétro-propagation d'erreur). Une autre raison est que la reconnaissance par l'approche symbolique en intelligence artificielle est limitée pour résoudre certains problèmes et que les réseaux de neurones fournissent une approche complémentaire à intégrer.

En géophysique, les applications de réseaux de neurones sont très nombreuses à savoir, la reconnaissance de forme en sismique (Plaza et Weger 1990), la combinaison de PS et le log de résistivité en vue d'estimer le log lithologique (McCormack 1991), l'interprétation des diagraphies (Wiener et al 1991, Lorenzetti 1992), la localisation d'un corps conductible à partir de champs électromagnétiques (Poulton et Al 1992) , la deconvolution (Wang et Mendel J. M. 1992), le picking des premiers arrivées (McCormack e Al. 1993), la caractérisation des réservoirs (An et Moon . 1993), l'analyse des composantes de l'onde S (Heng chang et Colin 1994), le picking dans l'enregistrement sismologique (Dai et MacBeth 1995), la correction dynamique (Calderon- Macias et Al. 1998), l'inversion en magnétotellurique (Spichak et Popova 2000) et l'inversion des données de résistivité (El-Qady et Ushijima 2001).

I.3 NEURONE BIOLOGIQUE

Dans le cerveau, les neurones sont reliés entre eux par l'intermédiaire d'axones et de dendrites (Fig1.1). En première approche, ces sortes de filaments sont des conducteurs d'électricité et peuvent ainsi véhiculer des messages depuis un neurone vers un autre. Les dendrites représentent les entrées du neurone et son axone sa sortie.
Un neurone émet un signal en fonction des signaux qui lui proviennent des autres neurones. Il est observé en fait au niveau d'un neurone, une intégration des signaux reçus au cours du temps, c'est à dire une sorte de sommations des signaux. En général, quand la somme dépasse un certain seuil, le neurone émet à son tour un signal électrique.
La notion de synapse explique la transmission des signaux entre un axone et une dendrite. Au niveau de la jonction (c'est à dire de la synapse), il existe un espace vide à travers lequel le signal électrique ne peut pas se propager. La transmission se fait alors par l'intermédiaire de substances chimiques, les neuromédiateurs. Quand un signal arrive au niveau de la synapse, il provoque l'émission de neuromédiateurs qui vont se fixer sur des récepteurs de l'autre côté de l'espace inter-synaptique. Quand suffisamment de molécules se sont fixées, un signal électrique est émis de l'autre côté et il y' a donc une transmission. En fait, suivant le type de la synapse, l'activité d'un neurone peut renforcer ou diminuer l'activité de ces voisins. On parle ainsi de synapse excitatrice ou inhibitrice.

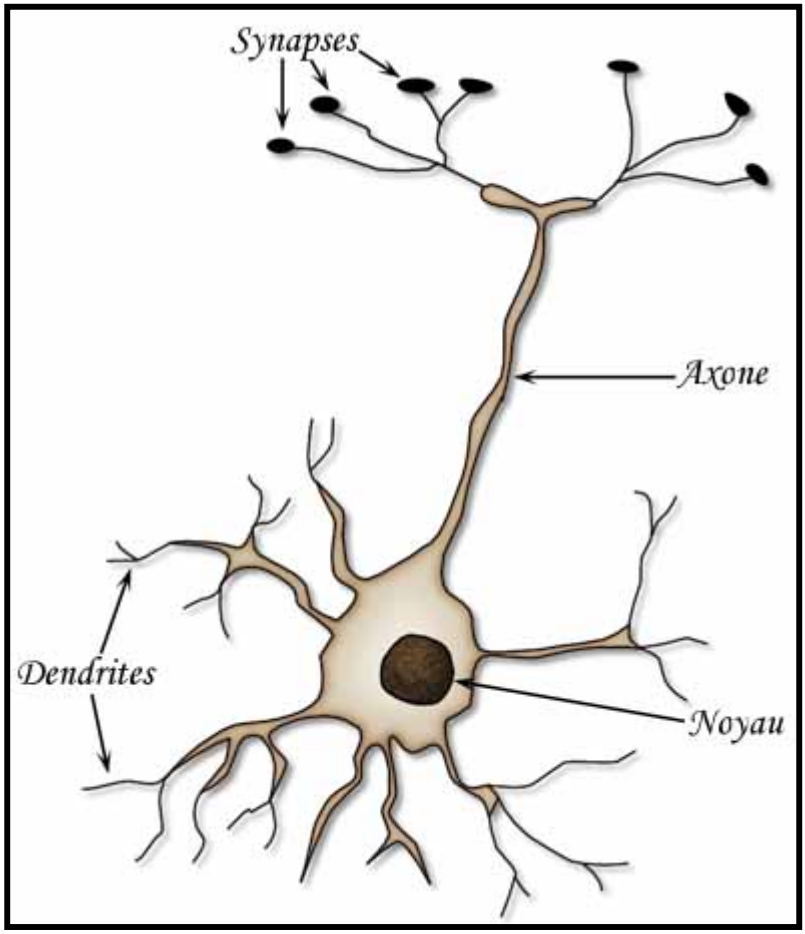


Fig.1.1 *Le neurone biologique*

I.4. RESEAUX DE NEURONES ARTIFICIELS

I.4.1. Neurone artificiel

Un réseau connexionniste est constitué d'éléments extrêmement simples qui interagissent pour donner au réseau son comportement global. Dans les modèles connexionnistes, ces éléments sont des processeurs élémentaires dont la définition est faite en analogie avec les cellules nerveuses, les neurones.
Ces unités de base reçoivent des signaux provenant de l'extérieur ou d'autres neurones du réseau. Ils calculent une fonction, simple en général, de ces signaux et envoient à leur tour des signaux vers un ou plusieurs autres neurones ou vers l'extérieur. La figure 1.2 montre un schéma comportant les éléments principaux d'un neurone artificiel.

Un neurone est caractérisé par trois concepts : son état, ses connexions avec d'autres neurones et sa fonction d'activation (Mejia, 1992).

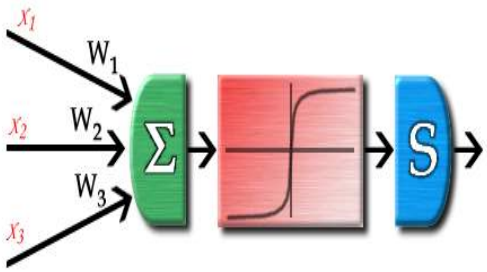


Figure 1.2 : *Neurone artificiel. L'état S du neurone est en fonction des entrées 1 , …, xn . Le neurone produit une sortie qui sera transmise aux neurones reliés.*

I.4.2. L'état des neurones

Un neurone artificiel est un élément qui possède un état interne. Il reçoit des signaux qui lui permettent éventuellement, de changer d'état. Nous noterons S l'ensemble des états possibles d'un neurone. S pourra être par exemple {0, 1} où 0 sera interprété comme l'état inactif et 1 l'état actif. S pourra également prendre un nombre plus grand de valeurs {0, 1, …, P}

Un neurone possède une fonction qui lui permet de changer d'état en fonction des signaux qu'il reçoit : c'est sa fonction d'activation.

L'état d'un neurone est fonction des états des neurones auxquels il est relié. Pour calculer l'état d'un neurone il faut donc considérer les connexions entre ce neurone et d'autres neurones. Nous définirons par la suite les connexions entre neurones et leur poids. Puis, nous parlerons du calcul de l'état d'un neurone.

I.4.3. Les connexions entre neurones

En 1943, Warren S. McCulloch et Walter Pitts ont proposé des réseaux appelés neuro-logiques, qui sont composés par l'interconnexion des petites unités élémentaires : les neurones formels.
Leur modèle était inspiré en partie des nouvelles théories mathématiques des automates à états finis de l'époque. Dans ce modèle, les neurones étaient arrangés d'une telle façon que le tout formait une machine capable, en particulier, de reconnaître des formes. L'organisation des neurones dans les réseaux, autrement dit l'architecture des réseaux, a été ici un facteur déterminant pour l'obtention de résultats intéressants.
L'architecture est le terme le plus général pour désigner la façon dont sont disposés et connectés les différents neurones qui composent un réseau. On parle également de topologie (terme emprunté de la théorie des graphes). Au niveau des neurones on parle plutôt de voisinage. Ce terme fait allusion à la façon dont un neurone est connecté à d'autres neurones. Il est donc en rapport direct avec l'architecture du réseau.

Le voisinage

Le voisinage d'un neurone est l'ensemble des neurones connectés à ce neurone. On parle de voisinage d'ordre n pour un neurone i, s'il y a n neurones connectés à ce neurone. Les connexions entre neurones ont souvent un sens.
Dans la figure 1.3, nous présentons des neurones avec des voisinages d'ordre 4, 8 et 6 respectivement. Les connexions utilisées ici n'ont pas de sens particulier, elles sont bidirectionnelles.

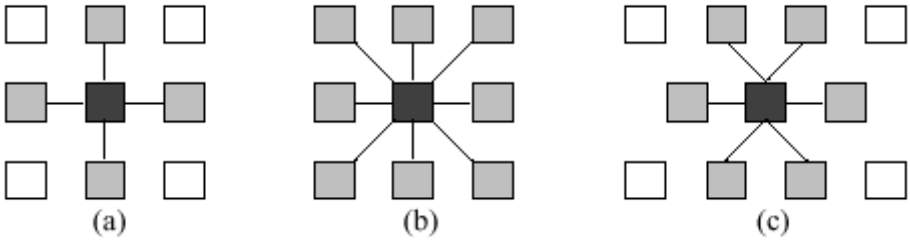


Figure 1.3 : *Plusieurs types de voisinage entre neurones. Les neurones en gris représentent le voisinage de celui du milieu.*

D'autres types de voisinage plus complexes sont possibles (Figure 1.4). On peut par exemple envisager des connexions complètes entre neurones, on aura alors un voisinage d'ordre N, N étant le nombre total de neurones du réseau (Figure 1.4.b).
Différents types de voisinage permettent de définir des architectures de réseaux de neurones différentes. Par exemple, la Figure 1.4.a montre un réseau où les neurones sont arrangés dans une grille. Ici les connexions reliant neurones ont un sens précis.

Les voisins les plus proches dans le sens horizontal ou vertical sont les seuls reliés au neurone du centre. La figure 1.4.c présente des neurones organisés par couches. Les connexions sont Exclusivement entre un ou plusieurs neurones appartenant à une couche du réseau et un neurone d'une couche différente. Il n'y a pas de connexions entre des unités appartenant à la même couche de neurones.

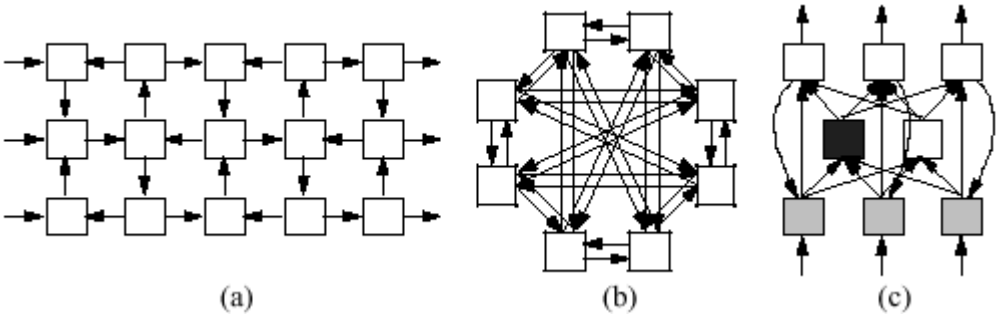


Figure 1.4 : *Organisation et voisinage dans un réseau de neurones. (a) les connexions du réseau sont partielles, les neurones sont arrangés dans une grille et ils ont des connexions exclusivement avec les voisins proches; (b) voisinage complet, chaque neurone dans le réseau est connecté avec la totalité des neurones du réseau; (c) les neurones sont organisés par couches, il y a des connexions seulement entre les neurones de couches différentes. En gris est signalé le voisinage du neurone en noir.*

Les connexions

Une connexion est un lien établi explicitement entre deux neurones. Les connexions sont aussi appelées *synapse*s, en analogie avec le nom des connecteurs des neurones réels.
On note dans la figure 1.4 que les liens entre neurones ont un sens, indiqué par une flèche (pour spécifier le sens bidirectionnel il faut définir deux connexions). Ce sens implique un flux d'informations donc, une dépendance. En effet, l'état d'un neurone *i* est fonction des états des neurones *j* connectés à *i* (flèches allant de *j* à i). De même, les états des neurones *l* auxquels *i* est connecté (flèches allant de *i* à l) sont influencés par l'état de i. Ceci spécifie encore plus notre notion de voisinage qui peut être défini finalement de la façon suivante :

$$i,j \in N, si \exists LIEN(j,i) \Rightarrow j \in VOISINAGE(i)... (1.1)$$

Où N est l'ensemble des neurones du réseau; *LIEN (j,i)* est une fonction booléenne qui est "vraie" si et seulement si le lien entre les neurones *j* et i, dans les sens *j*-i, existe; et *VOISINAGE(i)* représente l'ensemble des neurones connectés à i. Notez cependant que les neurones *l* auxquels *i* est connecté ne sont pas compris dans le voisinage de *i* à moins qu'il existe de façon réciproque un lien *l*-i (Mejia 3, 1992). Une connexion entre deux neurones a une valeur numérique associée appelé poids de connexion.

Les poids des connexions

Le poids de connexion *w ij* entre deux neurones *j* et *i* peuvent prendre des valeurs discrètes dans Z ou bien continues dans R. L'information qui traverse la connexion

sera affectée par la valeur du poids correspondant. Une connexion avec un poids $w\,ij = 0$ est équivalente à l'absence de connexion.
On définit une matrice des poids de connexions W où les lignes et les colonnes correspondent aux neurones et chaque valeur *w ij* représente le poids de la connexion entre la cellule *j* et la cellule *i* du réseau.

I.4.4 La fonction d'activation

Nous nous intéressons ici aux neurones qui calculent leur état à partir de l'information qu'ils reçoivent. Nous utiliserons par la suite la notation suivante :
S : l'ensemble d'états possibles des neurones.
x i : l'état d'un neurone i, où $x\,i \in$ S.
A i : l'activité du neurone i.
w ij : le poids de la connexion entre les neurones *j* et i.
L'activité d'un neurone est calculée en fonction des états des neurones de son voisinage et des poids de leurs connexions, selon la formule suivante :

$$A\,i = \sum_{j} w\,i\,j\,x\,j \ldots (1.2)$$

Comme il est illustré dans la figure 1.5 l'état *x i* du neurone *i* est une fonction de son activité *A i* :

$$x\,i = f(A\,i)\ldots (1.3)$$

La fonction f, appelée fonction d'activation peut avoir plusieurs formes et doit être dérivable. L'ensemble des états possibles dépend, bien entendu, de la fonction d'activation utilisée.

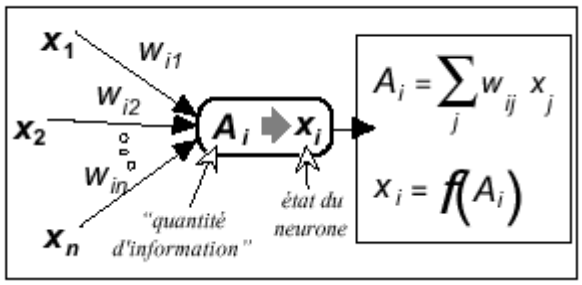


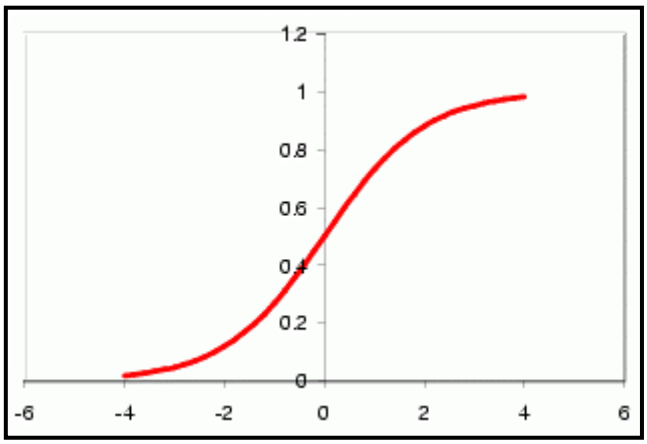
Figure 1.5 : *calcul de l'état d'un neurone. L'état x i d'un neurone i est une fonction des états des neurones j, de son voisinage, et des poids des connexions w* ij **.**

Mais le choix d'une fonction d'activation se révèle être un élément constitutif important des réseaux de neurones. Ainsi, l'identité n'est pas toujours suffisante et souvent des fonctions non linéaires et plus évoluées seront nécessaires.

A titre illustratif voici quelques fonctions couramment utilisées comme fonctions d'activation :

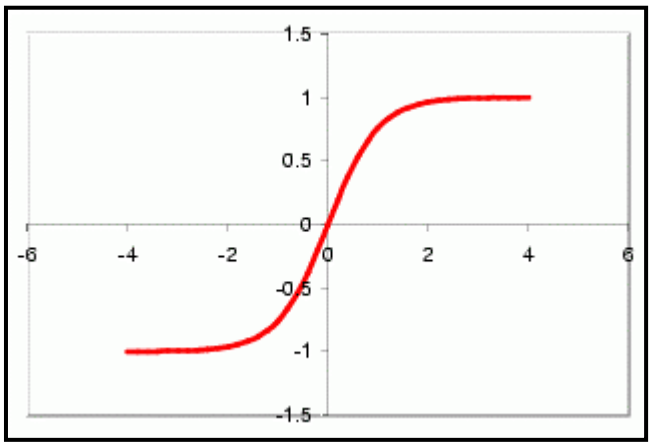
- ***Le sigmoid standard, (encore appelé fonction logistique) :***

F(X) = 1/(1 + exp(-d.X))



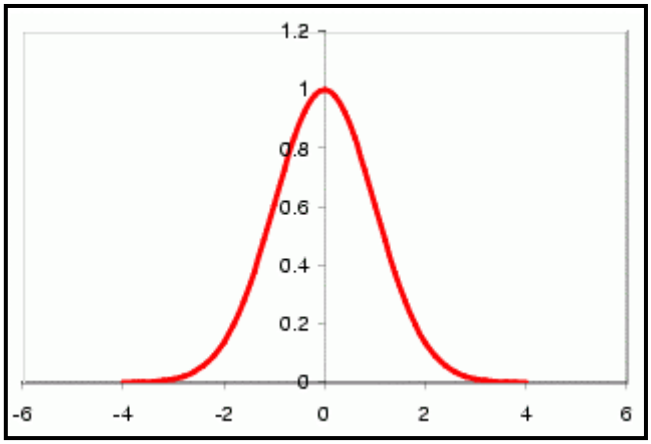
- ***La tangente hyperbolique :***

Y = 2 / (1 + exp(-2 . X)) - 1



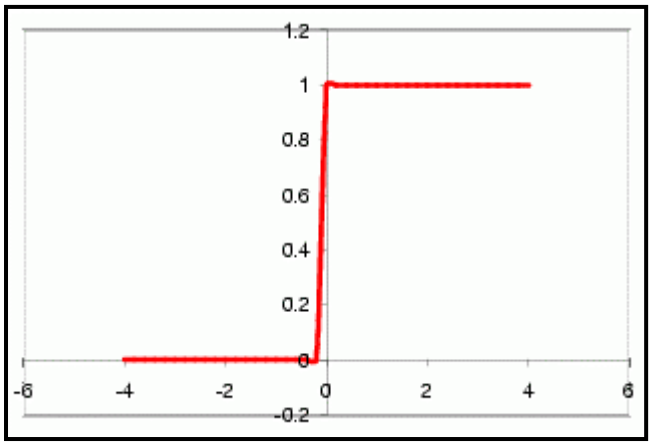
- ***La fonction Gaussienne :***

Y = exp(-x²/2)



- ***Une fonction à seuil :***

Y = 0 si X<0 et Y=1 si X≥0



I.4.5 Les architectures des réseaux de neurones

Introduction

Un réseau de neurones artificiels est un ensemble de neurones formels associés en couches et fonctionnant en parallèle.

Ces réseaux ont la capacité de stocker de la connaissance empirique et de la rendre disponible à l'usage. Les habiletés de traitement du réseau vont être stockées dans les poids synaptiques, obtenus par des processus d'adaptation

ou d'apprentissage. En ce sens, les RNA ressemblent donc au cerveau car non seulement, la connaissance est acquise à travers l' apprentissage mais de plus, cette connaissance est stockée dans les connexions entre les entités, soit dans les poids synaptiques. Les RNA peuvent être classées en deux grandes catégories:

a. Les réseaux « Feed-Forward »

Appelés aussi réseaux de type Perceptron, ce sont des réseaux dans lesquels l'information se propage de couche en couche sans retour en arrière possible.

a.1. Le Perceptron monocouche

C'est historiquement le premier RNA, c'est le Perceptron de Rosenblatt. C'est un réseau simple, puisqu'il ne se compose que d'une couche d'entrée et d'une couche de sortie. Il est calqué, à la base, sur le système visuel et de ce fait a été conçu dans un but premier de reconnaissance des formes. Cependant, il peut aussi être utilisé pour faire de la classification et pour résoudre des opérations logiques simples.

Sa principale limite est qu'il ne peut résoudre que des problèmes linéairement séparables. Il suit généralement un apprentissage supervisé selon la règle de correction de l'erreur.

a.2. Le Perceptron multicouches

C'est une extension du précédent, avec une ou plusieurs couches cachées entre l'entrée et la sortie. Chaque neurone dans une couche est connecté à tous les neurones de la couche précédente et de la couche suivante (excepté pour les couches d'entrée et de sortie) et il n'y a pas de connexions entre les cellules d'une même couche. Les fonctions d'activation utilisées dans ce type de réseaux sont principalement les fonctions à seuil ou sigmoïdes. Il peut résoudre des problèmes non-linéairement séparables et des problèmes logiques plus compliqués. Il suit aussi un apprentissage supervisé selon la règle de correction de l'erreur.

a.3. Les réseaux à fonction radiale

Ce sont les réseaux que l'on nomme aussi RBF « *Radial Basic Functions* ». L'architecture est la même que pour les Perceptrons multicouches cependant, les fonctions de base utilisées ici sont des fonctions Gaussiennes. Les RBF seront donc employés dans les mêmes types de problèmes que les PMC à savoir, en classification et en approximation de fonctions, particulièrement. L'apprentissage le plus utilisé pour les RBF est le mode hybride et les règles sont soit, la règle de correction de l'erreur soit, la règle d'apprentissage par compétition.

b . Les réseaux « Feed-Back"

Appelés aussi réseaux récurrents, ce sont des réseaux dans lesquels il y a retour en arrière de l'information.

b.1. Les cartes auto-organisatrices de Kohonen

Ce sont des réseaux à apprentissage non-supervisé qui établissent une carte discrète, ordonnée topologiquement, en fonction de patterns d'entrée. Le réseau forme ainsi une

sorte de treillis dont chaque noeud est un neurone associé à un vecteur de poids. La correspondance entre chaque vecteur de poids est calculée pour chaque entrée. Par la suite, le vecteur de poids ayant la meilleure corrélation, ainsi que certains de ses voisins, vont être modifiés afin d'augmenter encore cette corrélation.

b.2. Les réseaux de Hopfield

Les réseaux de Hopfield sont des réseaux récurrents et entièrement connectés. Dans ce type de réseau, chaque neurone est connecté à chaque autre neurone et il n'y a aucune différenciation entre les neurones d'entrée et de sortie. Ils fonctionnent comme une mémoire associative non-linéaire et sont capables de trouver un objet stocké en fonction de représentations partielles ou bruitées. L'application principale des réseaux de Hopfield est l'entrepôt de connaissances mais aussi la résolution de problèmes d'optimisation.

b.3 - Les ART

Les réseaux ART « *Adaptative Resonnance Theorie* » sont des réseaux à apprentissage par compétition. Le problème majeur qui se pose dans ce type de réseaux est le dilemme « stabilité/plasticité ». En effet, dans un apprentissage par compétition, rien ne garantit que les catégories formées vont rester stables.

La seule possibilité, pour assurer la stabilité, serait que le coefficient d'apprentissage tende vers zéro, mais le réseau perdrait alors sa plasticité. Les ART ont été conçus spécifiquement pour contourner ce problème. Dans ce genre de réseau, les vecteurs de poids ne seront adaptés que si l'entrée fournie est suffisamment proche, d'un prototype déjà connu par le réseau. On parlera alors de résonance. A l'inverse, si l'entrée s'éloigne trop des prototypes existants, une nouvelle catégorie va alors se créer, avec pour prototype, l'entrée qui a engendré sa création. Il est à noter qu'il existe deux principaux types de réseaux ART à savoir les ART-1 pour des entrées binaires et les ART-2 pour des entrées continues.

I.5. APPRENTISSAGE SUPERVISE

I.5.1. Introduction

L'objectif de l'apprentissage est de fournir une méthode au réseau afin qu'il puisse ajuster ses paramètres lorsqu'on lui présente des exemplaires à traiter.
Les poids obtenus sont déterminés par un algorithme dit d'apprentissage. Les algorithmes utilisés en connexionnisme fonctionnent à partir de règles mathématiques ou heuristiques qui modifient les poids itérativement quand on présente des exemples au réseau. On parle alors d'apprentissage à partir d'exemples. L'ensemble des données ayant servi à mettre au point les poids du réseau, est appelé ensemble d'apprentissage. Plusieurs quantités peuvent servir à mesurer la qualité des performances du réseau. Elles diffèrent selon la nature de la tâche que l'on veut faire réaliser au réseau.

Ces mesures peuvent être calculées sur l'ensemble d'apprentissage, toutefois, on est la plupart du temps intéressé non pas par ce type de performances mais par celles que le réseau obtiendra en phase opérationnelle sur des formes qui ne lui ont jamais été présentées. Pour estimer ces performances, on utilise la plupart du temps un ensemble de signaux dont on connaît les caractéristiques et qui n'ont pas servi à l'apprentissage. Cet ensemble est appelé ensemble test. D'autres ensembles de signaux sont quelquefois employés pour réaliser l'apprentissage, notamment pour faire de la validation croisée.
Pour une même architecture de réseau, on peut avoir différents algorithmes d'apprentissage qui diffèrent par la règle de mise à jour des poids.
On distingue deux grandes catégories de techniques d'apprentissage qui sont les algorithmes supervisés et ceux qui sont non supervisés.

Nous avons employé dans ce présent mémoire un algorithme supervisé, il s'agit de l'algorithme dit retro-propagation dont nous détaillons le principe.

I.5.2 L'algorithme de rétro-propagation du gradient

a. Introduction

Les limitations des modèles des réseaux de neurones des années 60 comme l'*Adaline* ou *le Perceptron*, ont conduit les chercheurs à abandonner progressivement cette ligne de méthodes d'apprentissage au profit des approches symboliques de ce que l'on appelle aujourd'hui l'Intelligence Artificielle classique.
Dans les années 80, grâce aux travaux de Teuvo Kohonen et de J. Hopfield , cette voie à été remise au goût du jour et a suscité un intérêt croissant de la part de nombreux chercheurs issus de différentes disciplines.

C'est ainsi que fut proposé simultanément par plusieurs équipes travaillant indépendamment l'algorithme de la rétro-propagation du gradient.
Cet algorithme peut être effectué en deux phases, à savoir :
- Alimentation avant *« Feed Foward »* : les sorties de neurone de la dernière couche sont calculées à partir des entrées de la première couche ;
- Rétropropagation de l'erreur : les poids sont ajustés pour que les sorties calculés et désirées se rapprochent le plus possible au sens de moindre carré.

b. Description de l'Algorithme

Notons x_i l'état d'un neurone i, $x_i \in \mathbf{S}$ qui est l'ensemble des états possibles.
L'état x_i du neurone *i* est calculé par :

$$x_i = f(A_i - \theta_i) \ldots (1.4)$$

où
f est une fonction sigmoïde ;
A i l'activité du neurone *i* ;
θ*i* le seuil associé au neurone i ;
Ai est l'activité donnée par :

$$Ai = \sum_j W_{ij} x_j \ldots (1.5)$$

où
j est indice des neurones "en amont" du neurone i ;
wij est le poids de la connexion du neurone *j* au neurone i.

On considère l'ensemble des vecteurs d'entrée x 1 , x 2 , …, x *m* définis dans R^n et les vecteurs réponses y 1 , y 2 , …, y *m* , définis dans R^p et W la matrice de poids de

Connexions définie dans R^{nc} , *nc* étant le nombre total de couches. Alors,

$x_k = [x_1^k, x_2^k, \ldots\ldots\ldots, x_n^k]$ sont les entrants.

$y_k = [y_1^k, y_2^k, \ldots\ldots\ldots, y_p^k]$ sont les sortants désirés

$$\mathbf{W} = \begin{bmatrix} 0 & 0 & \ldots & 0 & 0 \\ \mathbf{w}_1 & 0 & \ldots & 0 & 0 \\ 0 & \mathbf{w}_2 & \ldots & 0 & 0 \\ : & : & : & : & : \\ 0 & 0 & \ldots & \mathbf{w}_p & 0 \end{bmatrix} = \{w_{ij}\}_{nc \times nc'} \quad o\grave{u} \quad \mathbf{w}_i = \{w_{kl}\}_{nc_{(i-1)} \times nc_{(i)}}$$

Chaque sous-matrice **w** *i* contient les poids de connexions entre la couche i-1 avec *nc (*i-1) cellules et la couche *i* avec *nc (*i) cellules.

A chaque instant k, un vecteur x*k* est présenté dans la couche d'entrée du réseau. Les états des *n* cellules en entrée prennent les valeurs ($x_1^k, x_2^k, \ldots\ldots, x_n^k$).
Ces états sont propagés selon les équations (1.4) et (1.5), vers des unités se trouvant dans les couches en aval de la couche d'entrée jusqu'à arriver à la couche de sortie.

Appelons s*k* = ($s_1^k, s_2^k, \ldots\ldots, s_p^k$) la réponse du réseau pour une entrée x*k* donnée. s*k* est le vecteur composé par les états de *p* cellules de la couche de sortie.

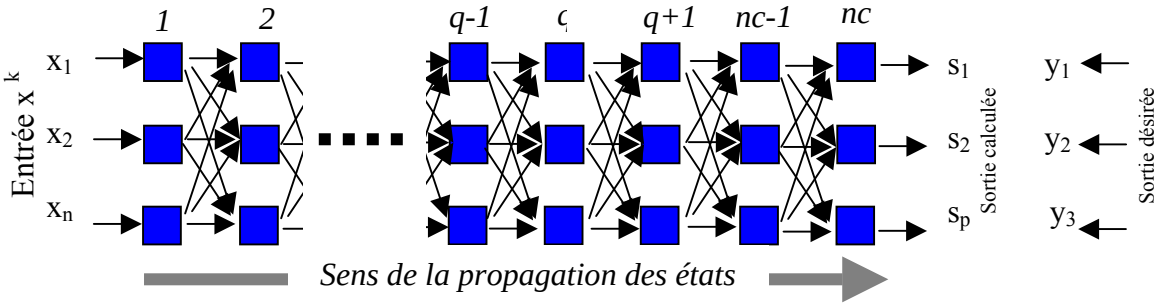


Figure 1.6 **:** *propagation des etats dans un reseau multicouche.La propagation desétats se fait "en aval" dès la couche d'entrée vers la couche de sortie. L'état de la cellule i dans la couche q est fonction des états des cellules j des couches précédentes q-1, etc.*

Le but étant d'obtenir pour une entrée x_k une réponse s_k la plus proche possible du vecteur y_k désiré correspondant, on est amené à optimiser notre réseau afin de réduire la différence entre sortie désirée et calculée.
Définissons la fonction objectif suivante :

$$C^k(W)=\sum_1^p (s_i^k - y_i^k)^2 \ldots (1.6)$$

qui est l'erreur ou distance quadratique entre la sortie calculée s_k et la sortie désirée y_k. La totalité des poids de connexions du réseau doit être ajustée en fonction de cette erreur. Cette fonction dépendra bien sûr de l'état du système et d'exemple d'apprentissage.

Pour minimiser la fonction objectif on doit calculer son gradient par rapport à W, alors l'équation générale d'adaptation des coefficients *w ij* est donnée comme suit :

$$w_{ij}^{k+1} = w_{ij}^k + \Delta w_{ij}^k \ldots (1.7)$$

dont les $\Delta w^k ij$ sont données par l'expression suivante :

$$\Delta w_{ij}^k = \varepsilon^k \frac{\partial C^k(\mathbf{W})}{\partial w_{ij}} = \varepsilon^k \delta_i^k x_j (1.8)$$

La valeur du pas d'apprentissage ε^k est typiquement assez petite et décroît dans le temps (Mejia, 1992).

Différents critères peuvent être employés pour arrêter le processus d'apprentissage.
Dans notre travail, nous avons élaboré un programme écrit en langage Fortran (voir annexe) permettant de réaliser le processus d'apprentissage. En plus, nous avons appliqué un logiciel pour le calcul des réseaux de neurones appelé Pythia. Dans Les sections qui suivent, nous présentons une brève description sur l'utilisation de ce soft.

I.6 APERÇU SUR LE LOGICIEL PYTHIA

Ce logiciel est conçu spécialement pour le calcul des réseaux de neurones artificiels. Les étapes principales de l'utilisation de ce logiciel sont les suivantes :

Etape 1 : Dans laquelle, les données d'apprentissages sont préparées dans une feuille Excel. Dans la figure (1.7), un exemple de données d'apprentissage est présenté. Le nombre de ces données est égal à quatre qui sont :

- $x_1(0,0)$, $y_1=0$; $x_2(0,1)$, $y_2=1$; $x_3(1,0)$, $y_3=1$; $x_4(1,1)$, $y_4=0$.

Nous devons mettre les entrants (x_1, x_2, x_3 et x_4) dans les deux premières colonnes et les sortants (y_1, y_2, y_3 et y_4) dans la dernière colonne.

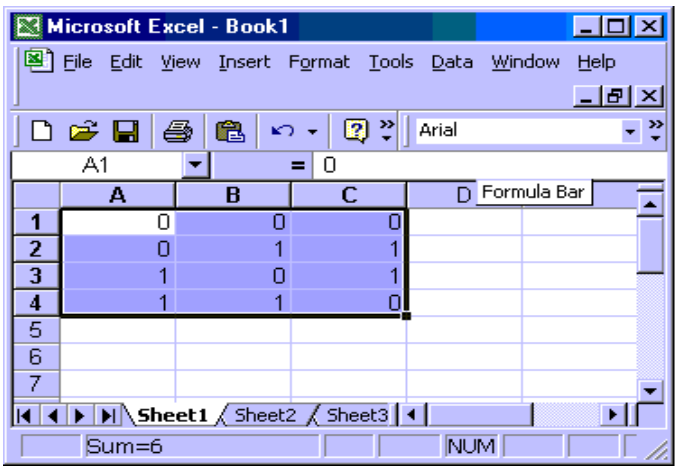


Fig1.7. *Préparation des données d'apprentissage.*

Etape 2 : Dans laquelle, le logiciel Pythia est exécuté et les données qui sont déjà préparées dans la feuille Excel sont insérées dans la feuille Pythia, on aura la fenêtre montrée dans la figure (1.8).

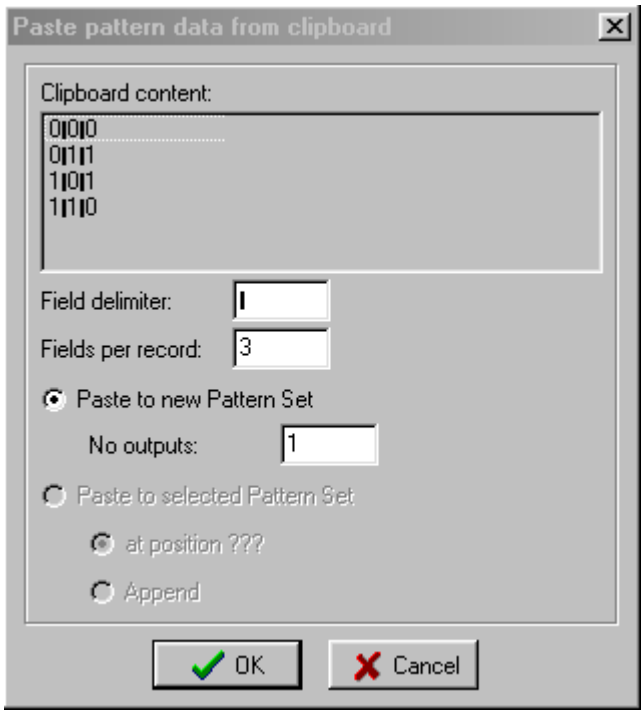


Fig1.8. *Insertion des données d'apprentissage*

Tels que :

Field per record : est la somme de la dimension des entrants (vecteurs à deux dimensions) et la dimension des sortants (vecteurs à une seule dimension).

No Output : est la dimension des sortants (vecteurs à une seule dimension).

Etape 3 : Dans laquelle on définit la structure de réseau qui nous intéresse. En cliquant sur « Create New Net », on aura les trois figures ci-dessous :

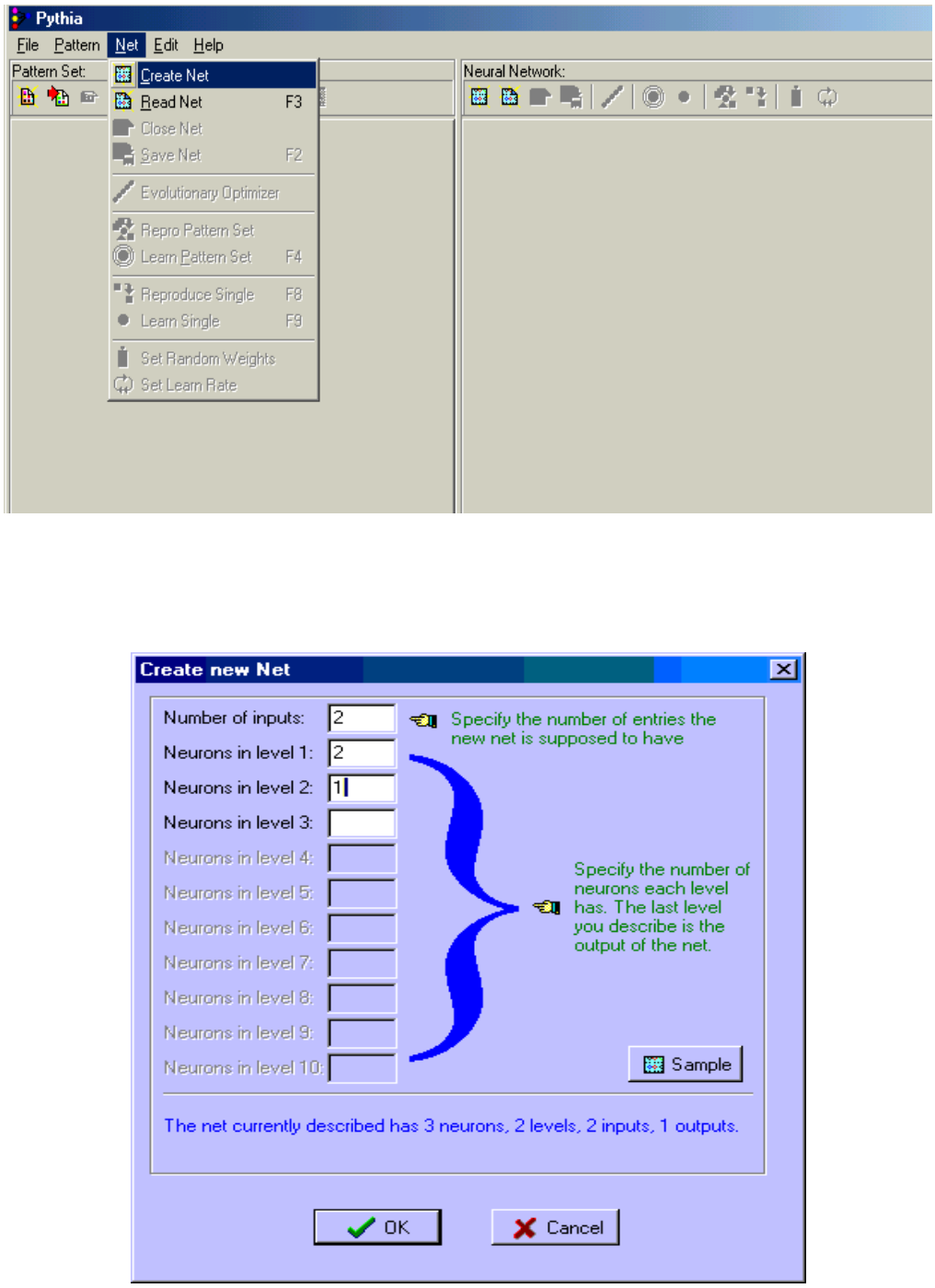


Fig. 1.9 *Choix des paramètres de réseau*

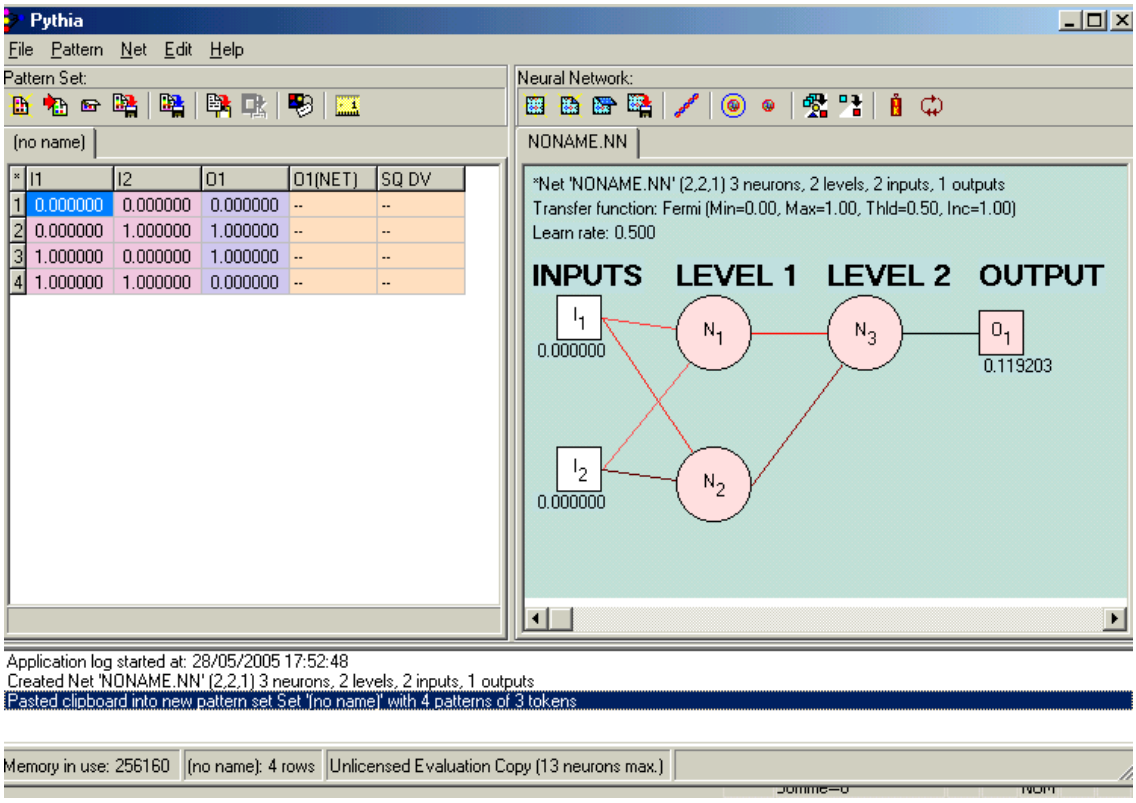


Fig1.10. *Structure de réseau de neurone*

Etape 4 : Après avoir inséré les données d'apprentissage et choisit la structure du réseau, on arrive à l'étape d'apprentissage, et ce en cliquant sur Net dans la barre de titre puis sur « Learn Pattern Net », on aura la figure ci-dessous :

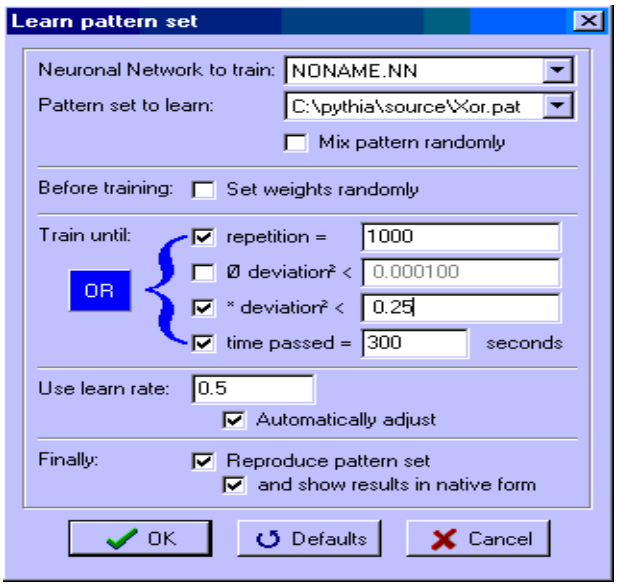


Fig. 1.11. *L'étape d'apprentissage*

Alors, les poids synaptiques vont être affichés automatiquement dès que l'étape d'apprentissage est terminée et ce, en mettant le curseur sur le neurone dont on désire afficher les poids.

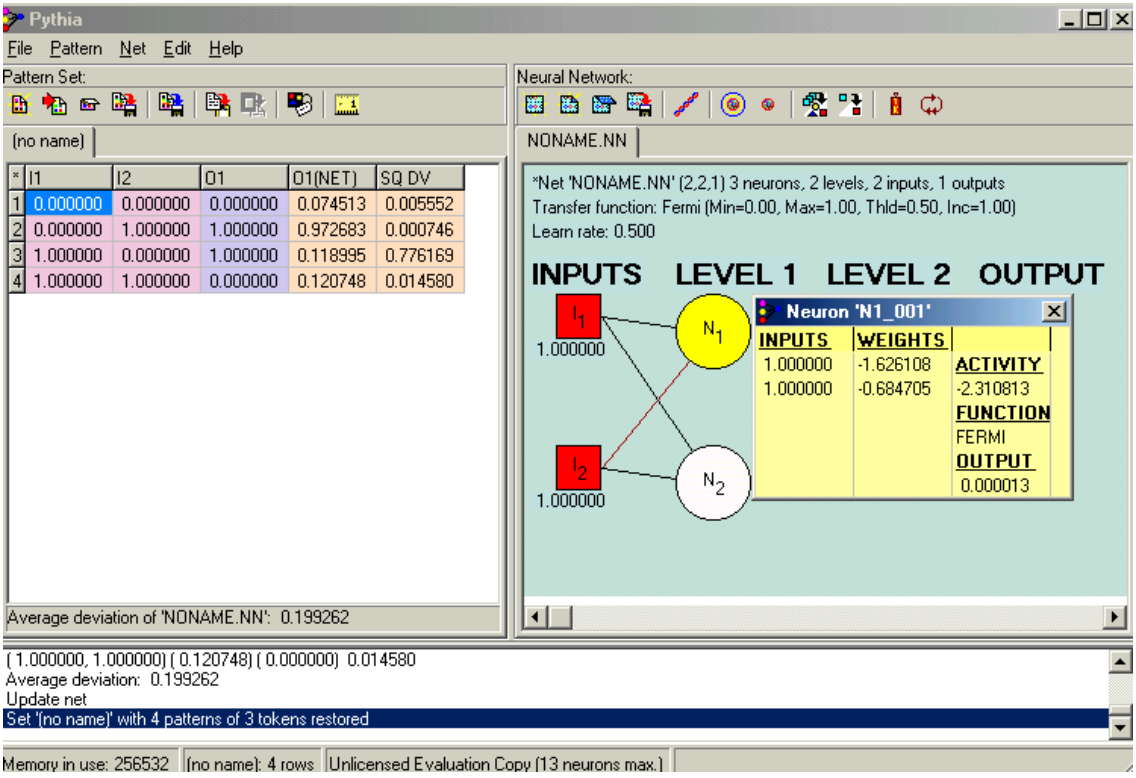


Fig. 1.12. *Poids synaptiques de réseau*

Chapitre II

II.1. INTRODUCTION

En sismique, l'étape du pointé des horizons sismiques est la plus importante dans la phase d'interprétation. En effet, le suivi de ces marqueurs fournit une image structurelle de sous-sol .

A cet effet, nous allons présenter deux techniques pour le pointé des horizons sismiques. Il s'agit d'une technique dite moments invariants et d'une autre basée sur l'intelligence artificielle.

II.2. GENERALITES SUR LE POINTE DES HORIZONS SISMIQUES.

II.2.1. Introduction

La méthode sismique est essentiellement constituée de trois étapes à savoir :

- L'acquisition des données aux missions sismiques ;
- Le traitement des données ;
- L'interprétation structurale ou stratigraphique des sections sismiques qui offrent aux géologues des éléments permettant de comprendre l'histoire d'un bassin ou d'un gisement et sa formation.

L'interprétation structurale des sections sismiques est faite selon les étapes suivantes :

- Le choix des horizons ;
- Le pointé (pointé des arrivées réfléchies) ;
- Le chronométrage ;
- Le report des valeurs lues sur le plan de position ;
- Le tracé des différentes cartes ;
- L'interprétation et la conclusion.

Le pointé des réflexions qui correspondent à l'horizon ciblé est l'étape la plus délicate et la plus importante, elle conditionne la fiabilité des résultats obtenus. Cette étape est réalisée selon trois aspects à savoir :

- Le pointé manuel ;
- Le pointé automatique ;
- Le pointé par réseaux de neurones.

L'identification des arrivées sismiques est basée sur différents critères à savoir, la cohérence, la variation de l'amplitude, le caractère, la courbure due au pendage et la courbe normale (Benhama, 2000).

La cohérence est le critère le plus important. Si l'onde atteint le dispositif, elle produit le même effet, c'est cette similitude qui est appelée la cohérence.

La variation de l'amplitude est un critère en relation avec l'accroissement de l'amplitude des traces résultant de l'arrivée d'une énergie en phase sur les différents canaux.

Le caractère permet de discerner une arrivée donnée. Il concerne la forme de l'enveloppe, le nombre de cycles pendant lesquels l'amplitude augmente et les irrégularités de phase provenant d'interférences entre les composantes de l'arrivée.

Les problèmes rencontrés lors de l'interprétation se résument en :
- Dédoublement d'un horizon : qui peut être dû à un épaississement de la série, à un changement de faciès, aux interférences (multiples, bruits organisés, hyperboles de diffraction), mauvaises corrections statiques ;
- Variation spatiale du rapport signal sur bruit ;
- La variation définie par le contraste d'impédance entre deux niveaux, appelée la discordance.

En présence de ces contraintes, les critères de tracé des horizons demeurent moins efficaces. Pour cela, d'autres techniques basées sur d'autres critères ont été développées. Il s'agit des techniques de pointé automatique.

II.2.2. Définition
En général, le pointé d'un événement consiste à classifier tous les événements sur une matrice de données ayant un ou plusieurs critères communs.
Le pointé automatique de l'horizon revient à pointer les temps d'arrivée de la même arrivée réfléchie choisie sur une trace initiale le long du profil sismique et ayant les mêmes attributs sismiques ou autres critères de définition.

II.3. PROBLEMATIQUE DE L'ESTIMATION DE DELAI
II.3.1 Introduction
L'estimation de délai est un problème soulevé dans différentes disciplines (le radar, l'imagerie médicale, la géophysique..). L'objectif est de mesurer le décalage entre deux signaux.
Dans la sismique, deux cas d'estimation peuvent se présenter :
- Estimation du délai sur un point de tir ;
- Estimation du délai sur une section stack.

Dans notre cas, nous avons utilisé deux techniques, il s'agit d'une technique de corrélation appelée moment invariant et une technique basée sur l'intelligence artificielles. Les deux techniques ont été appliquées sur des données sismiques réelles.

II.3.2 Définition du délai
Le délai est définit par le retard entre deux mesures quelconques. Dans la figure (2.1), les deux récepteurs R1 et R2 reçoivent le signal émis par la source O à des temps différents. Ce retard est dû principalement à la disposition des récepteurs, à la distorsion du signal émis et aux bruits additifs sur les enregistrements.

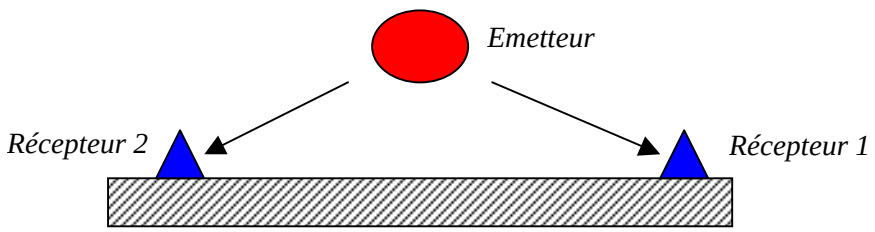
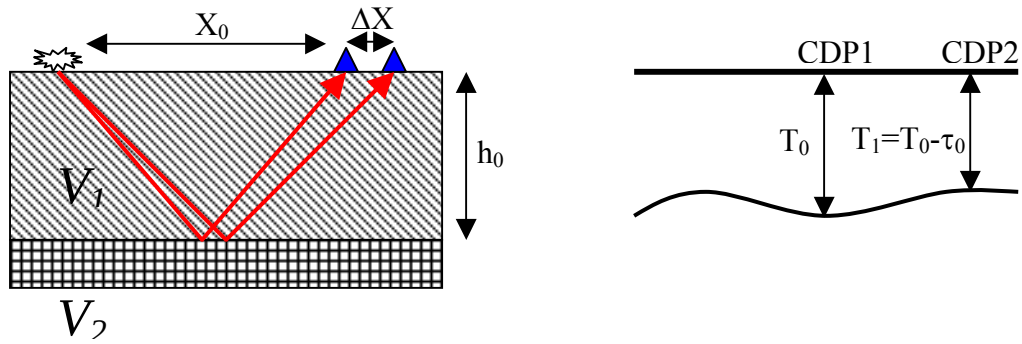


***Figure 2.1** : Définition du délais*



a. *Point de tir* ***b.*** *Section sismique*

Figure 2.2: *Modèle du délai sismique.*

Dans la sismique, le délai sur enregistrement point de tir (figure 2.2.a) est calculé par la formule suivante :

$$\Delta t = T_{R2} - T_{R1} = \frac{V_1(\Delta X^2 + X_0^2)}{4 h_0 V_2^2}$$

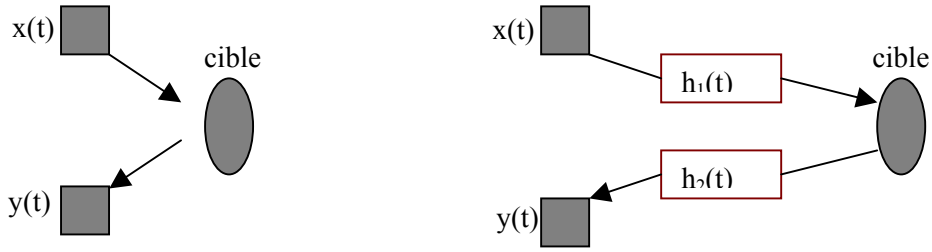
Où Δt représente le décalage en temps de la même réflexion.

Dans la section sismique (figure2.2.b), le retard τ_0 peut être dû à une courbure de l'horizon, à une variation spatiale de vitesse, à une mauvaise correction statique et à une mauvaise analyse de vitesse.

II.3.3 Formulations mathématiques

Formulation générale

Dans cette section, deux modèles mathématiques sont présentés, il s'agit d'un modèle simpliste et un deuxième distordu (Guerchaoui, 1999), qui sont représentés dans le schéma suivant :



a. *modèle simple* ***b.*** *modèle distordu*

Figure 2.3 : *Génération des systèmes à délai.*

Modèle simpliste (figure 3.2.a) :
Ce modèle est décrit mathématiquement par les équations suivantes :

$$x(t)= s(t)+n_1(t) \ldots (2.1)$$
$$y(t)=s(t-d)+n_2(t) \ldots (2.2)$$

Où
s(t) est le signal inconnu et s(t-d) sa version décalée.
$n_1(t)$ et $n_2(t)$ sont des bruits additifs.

Modèle distordu (figure 3.2.a) :
Ce modèle est décrit mathématiquement par les équations suivantes :

$$x(t)=h_1(t)*s(t)+n_1(t) \ldots (2.3)$$
$$y(t)=h_2*s(t-d)+n_2(t) \ldots (2.4)$$

Où
$h_1(t)$ et $h_2(t)$ représentent les distorsions des deux systèmes linéaires.

Formulation sismique
Avant de présenter cette formulation, on doit tout d'abord présenter un aperçu sur la modélisation d'une trace sismique.

Le signal sismique est le résultat des arrivées en surface d'ondelettes à des temps différents avec des amplitudes différentes.
La trace sismique peut être modélisée par une série de processus de convolution en cascade (domaine temporel).
Le modèle de la trace sismique proposé par Telford et al. 1976 (Telford et al., 1998) est :

$$T(t)=s(t)*EAR(t)*WZ(t)+B(t) \ldots (2.5)$$

Où
T(t) est la trace sismique enregistrée ;
S(t) le signal émis par la source sismique ;
EAR(t) la réponse des terrains avec des phénomènes d'atténuation ;
WZ(t) l'effet de la zone altérée (ou des réverbérations dans la couche d'eau pour le cas de la sismique marine) ;
B(t) l'ensemble des bruits divers. En général, B(t) est aléatoire et blanc ;
* la convolution temporelle ;
EAR(t) est donnée par la relation suivante :

$$EAR(t)=A(t)*M(t)*R(t)*DS(t) \ldots (2.6)$$

Où
A(t) est l'atténuation des couches géologiques (absorption) ;
M(t) l'atténuation due aux réflexions multiples et aux effets de dispersion dans le sous sol ;
R(t) la série des coefficients de réflexion ;
DS(t) la divergence sphérique.

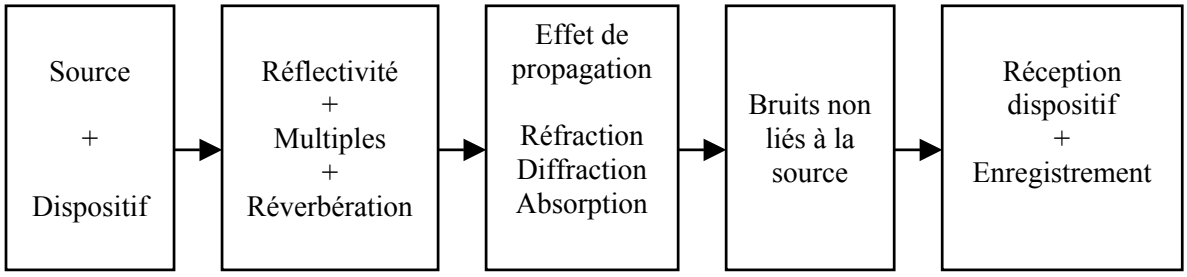


Figure2.4 : *Modèle complet de la trace sismique*

En sismique réflexion, x(t) et y(t) représentent deux traces voisines, s(t) la signature source de l'émission, $h_1(t)$ et $h_2(t)$ les distorsions correspondantes.

On cherche à estimer le délais τ_0 dans une situation de méconnaissance totale ou partielle de ces distorsions et des caractéristiques des bruit $n_1(t)$ et $n_2(t)$
Dans les conditions d'un bon traitement des données des deux enregistrements, on aura :

$$n_1(t) \approx n_2(t) \approx 0 \ldots (2.7)$$

Alors, les deux équations (2.3) et (2.4) deviennent :

$$x(t)=h_1(t)*s(t) \ldots (2.8)$$
$$y(t)=h_2*s(t-\tau_0) \ldots (2.9)$$

Dans le cas où la distorsion des deux signaux est négligeable, on aura :

$$x(t)=s(t) \ldots (2.10)$$
$$y(t)=s(t-\tau_0) \ldots (2.11)$$

Dans ce cas, le délai τ_0 est facile à estimer, il se résume à pointer les deux maxima de l'énergie de x(t) et y(t) et d'évaluer la différence du temps.
La problématique se l'estimation du délai se réduit à la définition du signal s(t). Autrement dit, à localiser l'image de ce signal à partir de son original s(t).

II.4. TECHNIQUES DE POINTE DES HORIZONS SISMIQUES.

Dans cette section, nous allons présenter deux techniques d'estimation de délai afin de pointer les horizons sismiques.

II.4.1. Technique des moments invariants

C'est une technique qui appartient aux méthodes de corrélation, son principe revient à caractériser une image à une seule dimension par un vecteur de moments invariants d'une dimension donnée ; il consiste à trouver l'image modèle de la trace à traiter à partir de l'image de la trace précédente.

Moments invariants

Soit x(i) ; i=1,n les échantillons de la trace à traiter.
Le vecteur des moments centrés du signal x(i) pour un ordre n, calculé sur une fenêtre de longueur L, est donné par la relation suivante :

$$\mu_i=\{\mu_1,\mu_2,.....,\mu_n\}\ldots(2.12)$$

Tels que :

$$\mu_0=\frac{1}{n}\sum_{i=1}^{n}x(i)$$

$$\mu_1=\frac{1}{n}\sum_{i=1}^{n}i(x(i)-\mu_0)$$

$$\mu_2=\frac{1}{n}\sum_{i=1}^{n}i^2(x(i)-\mu_0)^2$$

$$\mu_3=\frac{1}{n}\sum_{i=1}^{n}i^3(x(i)-\mu_0)^3$$

……..

$$\mu_n=\frac{1}{n}\sum_{i=1}^{n}i^n(x(i)-\mu_0)^n$$

Dans cette technique, on calcule le vecteur des moments invariants noté Vt de la trace à traiter, puis on calcule le même vecteur noté Va de la trace adjacente sur une fenêtre glissante de longueur L où le glissement de cette dernière se fait par pas d'échantillonnage, et la longueur maximale décrite sur la trace étant de (2c+1).l avec 0<c<1, où c est le taux d'ajustement de la fenêtre.
La position de réflecteur est choisie selon le critère suivant (Guerchaoui et Benhama, 2000):

$$||Va-Vt||^2 \text{ minimum}\ldots(2.13)$$

Cette technique peut être résumée dans l'organigramme ci -après :

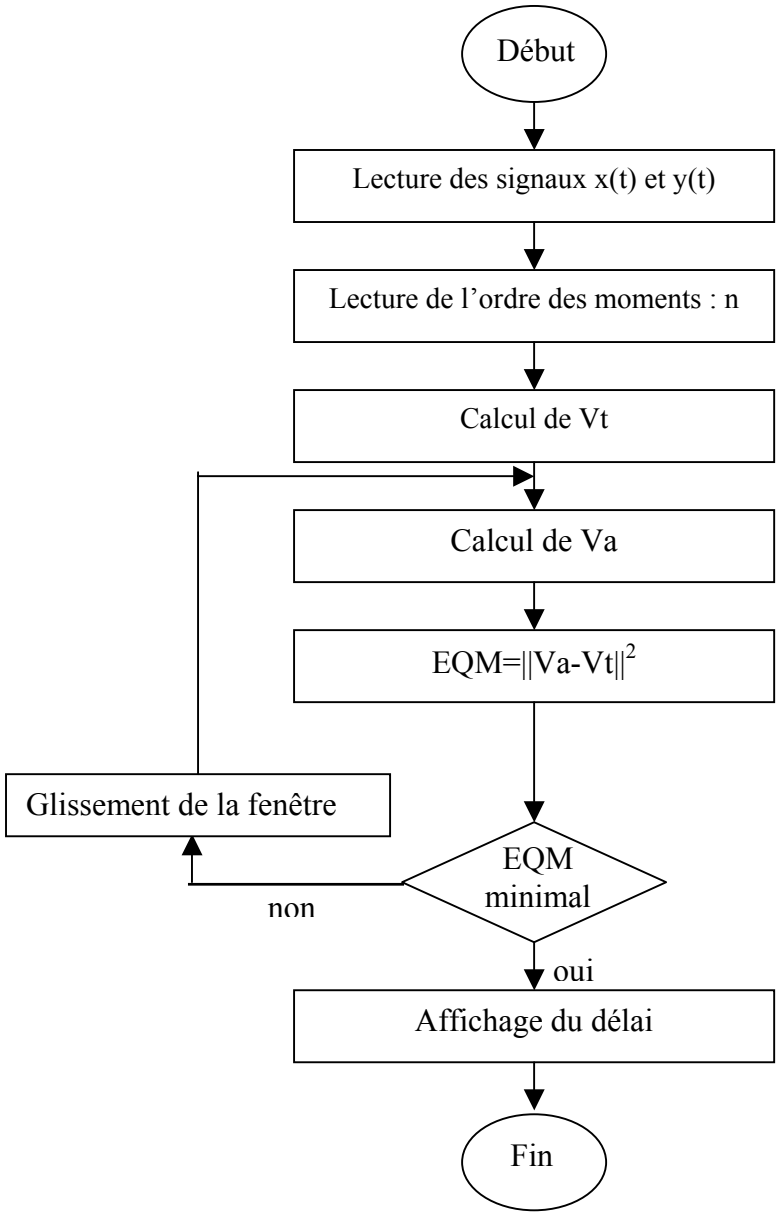


***Figure 2.5**: Organigramme de la technique des moments invariants*

II.4.2. Technique d'intelligence artificielle

II.4.2.1 Introduction

Dans le domaine pétrolier, les réseaux de neurones peuvent résoudre une multitude des problèmes rencontrés dans l'analyse des images sismiques, l'analyse de la productivité d'un puits, et l'analyse des propriétés des roches.

Dans ce présent mémoire, nous avons appliqué cette fameuse technique pour le pointé des horizons sismiques lors de la phase d'interprétation.

Les réseaux de neurones sont des systèmes dynamiques non linéaires, capables d'identifier des modèles très complexes après avoir subi un processus d'apprentissage.

Cette d'étape d'apprentissage nécessite un choix judicieux des données entrants et des données sortants.
Pour le suivi automatique des horizons sismiques, les entrants utilisés sont les attributs sismiques des segments de l'horizon ciblé, et la sortie du réseau utilisé est représentée par un coefficient de ressemblance normalisé à l'unité (Ulrich , 1999).

II.4.2.2 Topologie de RNA

Pour le pointé des horizons sismiques, nous avons choisi l'architecture suivante :

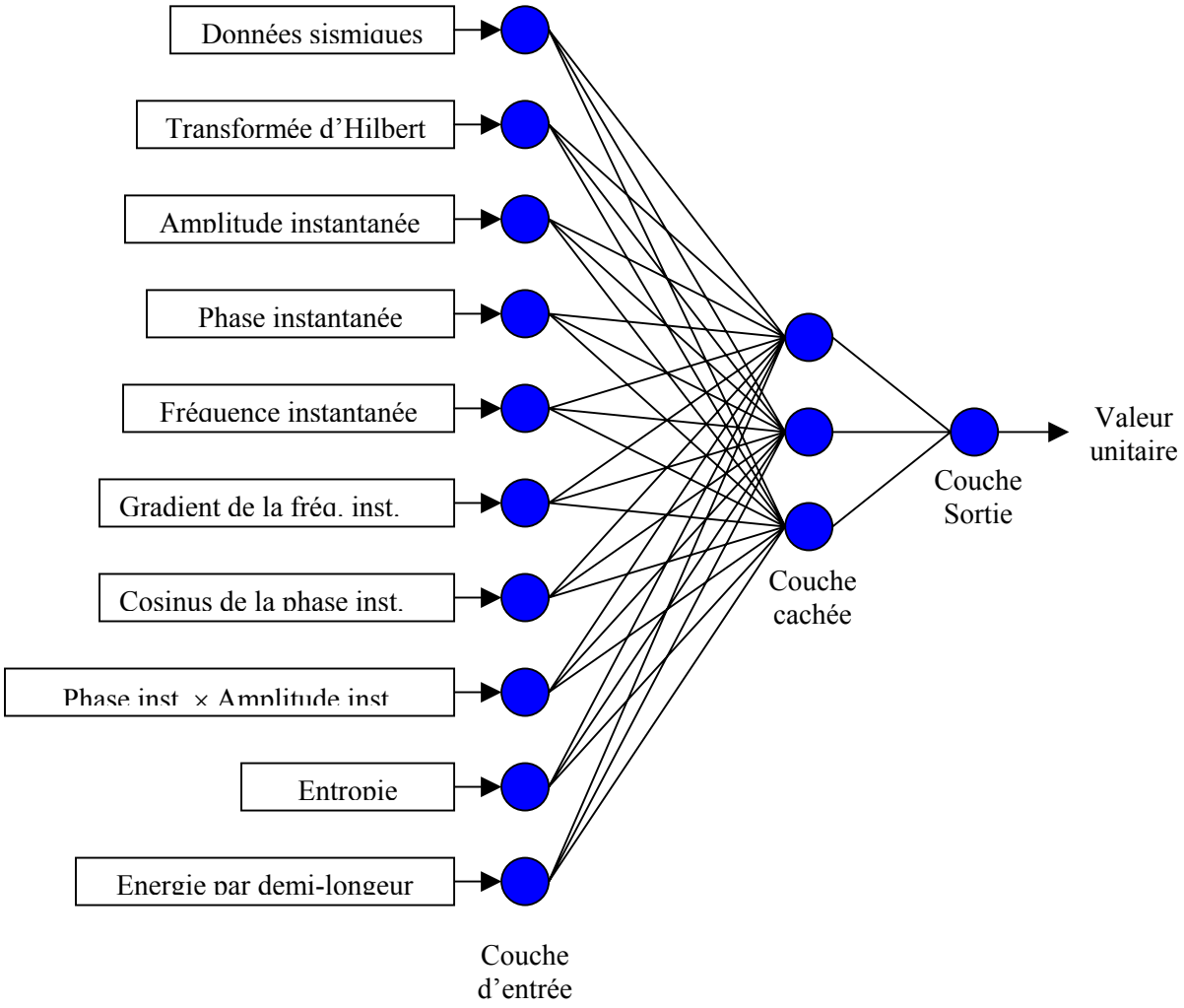


Figure 2.6 : *Topologie du RNA*

II.4.2.3 Etape d'apprentissage

L'apprentissage est appliqué sur plusieurs portions de la section sismique afin d'obtenir le résultat le plus fiable et ceci en utilisant l'algorithme de retro-propagation.
En premier lieu, les poids du RNA notés w_{ij} et w_j sont initialisés selon les relations suivantes :

$$-0.5< w_{ij}<0.5$$

$$-0.5 < w_j < 0.5$$

En deuxième lieu, l'algorithme de retro-propagation est appliqué afin d'arriver à une convergence et une stabilité des coefficients w_{ij} et w_j qui seront sauvegardés.

II.4.2.4 Etape de suivi

Cette étape consiste à appliquer sur une fenêtre glissante le RNA obtenu par apprentissage, et choisir la fenêtre qui donne la valeur la plus proche à l'unité. Cette fenêtre correspond bien à l'horion ciblé initialement.

Chapitre III

III.1 NTRODUCTION

L'inversion électrique a été reconnue depuis longtemps comme un problème non linaire ou quasi non linéaire. Les méthodes linéaires itératives sont tributaires d'un bon choix du modèle initial. Par contre, la plus part des méthodes non linéaires ne dépendent pas du modèle initial, mais un bon choix de ce dernier réduit énormément le temps du calcul (Chunduru et al., 1996).

Dans notre travail, nous avons utilisé les RNA pour l'interprétation du sondage électrique. Dans les sections suivantes, nous détaillons la théorie du sondage électrique.

III.2. LE SONDAGE ELECTRIQUE

III.2.1 Introduction

Le sondage électrique a été appliqué pour la première fois par Conrad Schlumberger en 1912 dans le but d'avoir la variation de la résistivité en fonction de profondeur ($\rho = f(p)$) (Koefoed, 1997). Dans la phase de l'interprétation, le sous-sol est supposé constituer de plusieurs couches horizontales et électriquement homogènes et isotropes.

III.2.2 Configuration du dispositif de mesure

a. Dispositif de Schlumberger

La configuration la plus utilisée est celle de Schlumberger (figure3.1), dans laquelle quatre électrodes placées d'une manière symétrique, deux électrodes à l'extérieur assurant l'émission du courant, et deux autres à l'intérieur mesurant la différence de potentiel.

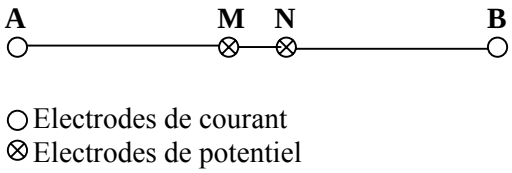


Figure 3.1 Dispositif de Schlumberger

La ligne de réception MN a une longueur toujours plus faible vis-à-vis de la ligne d'émission AB. On lui donne une valeur telle que V soit mesurable avec les appareils dont on dispose (Lasfargues, 1957)

b. Dispositif de Wenner

Une autre configuration qui a été développée quelques années plus tard au Etats Unis par Wenner (figure3.2), dans laquelle la distance entre les électrodes est la même.

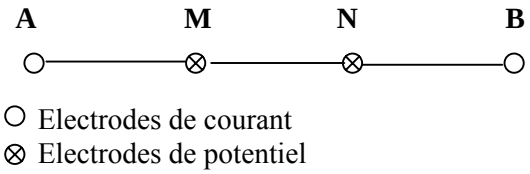


Figure 3.2 Dispositif de Wenner

Différence entre les deux dispositifs (Schlumberger et Wenner)

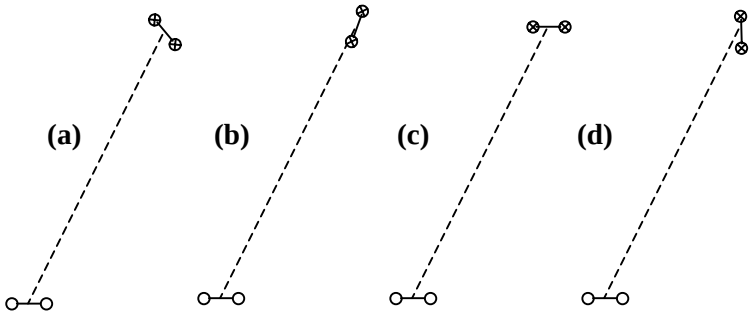
Dans les deux dispositifs, la profondeur de pénétration est contrôlée par la distance entre les électrodes de courant.
Par contre la différence de potentiel mesuré est plus importante dans le dispositif de Wenner que dans celui de Schlumberger, en effet la précision de mesure est plus grande dans le premier dispositif que dans le second.
L'avantage majeur de dispositif de Schlumberger est que l'erreur produite par les hétérogénéités latérales superficielles est systématique, et pourra être détectée à l'interprétation, ce qui n'est pas le cas dans le dispositif de Wenner.

c. Configuration dipolaire

Ce type de dispositif a été inventé en ex- URSS (figure3.3). Cette configuration a les caractéristiques suivantes :

- La distance entre les deux électrodes de courant ou de potentiel est très petite ;
- La distance entre les dipôles est très grande permettant de contrôler la profondeur de pénétration.

Dans ce dispositif, on distingue quatre types de configuration.



(a) azimutal
(b) radial
(c) parallèle
(d) perpendiculaire

II.2.3 Choix de site de mesure

La position des électrodes de potentiel doit être la plus loin possible de toute hétérogénéité latérale qui se trouve à la surface et qui pourrait falsifier les mesures (Koefoed, 1979).

II.2.4 Courant de circuit

Le courant utilisé dans le sondage électrique est un courant continu, si le courant est alternatif, le modèle de flux de courant est complètement changé ; ceci est dû à l'effet de peau, et dans ce cas, la densité de courant décroît selon l'équation suivante :

$$j_z = j_o e^{-\frac{z}{\xi}} \quad \text{....(3.1)}$$

Où

j : La densité de courant.

Z : La profondeur.

ξ : La profondeur de pénétration, donnée par :

$$\xi = \sqrt{\frac{\rho}{\pi \mu f}} \quad \text{....(3.2)}$$

où

ρ : La résistivité du sol.

μ : La perméabilité magnétique du sol.

f : La fréquence du courant.

On déduit que l'effet du courant alternatif sur la distorsion du flux du courant est négligeable pour les fréquences très faibles.

Il peut arriver que la résistivité de la couche superficielle soit très élevée. Alors pour assurer une bonne émission de courant on procède de deux manières :

- Mouiller la surface par l'eau salée « *salt water* » ;

- Utiliser plusieurs électrodes connectées parallèlement.

II.2.5 Distribution de potentiel à la surface

a. Introduction

En 1930 Stefanescu et al. ont pu calculer le potentiel à la surface, avec les hypothèses suivantes :

- Le courant émis par la source est un courant continu ;
- Le champ est créé par une source ponctuelle de courant localisée à la surface ;
- Le sous-sol est constitué de plusieurs couches horizontales dont la dernière est étendue à l'infini ;
- Chaque couche est électriquement homogène et isotrope.

b. Solution générale du champ de potentiel.

Le potentiel V satisfait l'équation de LAPLACE donnée par :

$$\frac{\partial^2 V}{\partial^2 x}+\frac{\partial^2 V}{\partial^2 y}+\frac{\partial^2 V}{\partial^2 z}=0 \dots (3.3)$$

Dans les cordonnées cylindriques cette équation devient :

$$(\frac{\partial^2}{\partial r^2}+\frac{\partial}{r\partial r}+\frac{\partial^2}{\partial z^2})V=0 \dots (3.4)$$

Supposant qu'il existe des solutions de la forme : v(r,z)=u(r).w(z), on aura le système d'équation suivant :

$$\begin{cases} \frac{\partial^2 u}{v\partial^2 u}+\frac{\partial^2 u}{ur\partial^2 r}=-\lambda^2 \dots (3.5.a) \\ \frac{\partial^2 w}{w\partial^2 z}=\lambda^2 \dots (3.5.b) \end{cases}$$

Où λ est une constante réelle arbitraire.

La solution de l'équation (3.5.a) est donnée par : $w=ce^{-\lambda z}$ et $w=ce^{+\lambda z}$

La forme de l'équation (3.5.b) a incité les matheux à créer une classe de fonction appelée la fonction de Bessel, et la solution est donnée par : $u=cj_0(\lambda r)$.

Où j_0 est la fonction de Bessel d'ordre zéro.

Combinons les solutions précédentes, nous aurons :

$$V= ce^{-\lambda z} j_0(\lambda r) \qquad \text{et} \qquad V= ce^{+\lambda z} j_0(\lambda r) \quad \ldots.(3.6).$$

Mettons λ varie de zéro à l'infini et C dépendant de λ, on obtient la solution générale suivante :

$$V = \int_0^{\infty} \left[\Phi(\lambda)e^{-\lambda z} + \Psi(\lambda)e^{+\lambda z}\right] j_0(\lambda)d\lambda \quad \ldots(3.7).$$

Où Φ(λ) et Ψ(λ) sont des fonctions arbitraires de λ, qui sont déduites à partir des conditions aux limites.

Alors le potentiel généré par une source ponctuelle située à la surface d'un sous-sol électriquement homogène est donné par :

$$V = \frac{\rho I_1}{2\pi\sqrt{r^2 + z^2}} \ldots.(3.8)$$

Utilisons l'intégral de Lipschitz donné par :

$$\int_0^{\infty} e^{-\lambda z} j_0(\lambda r)d\lambda = \frac{1}{\sqrt{r^2 + +z^2}}$$

Le potentiel peut être écrit comme suit :

$$V = \frac{\rho_1 I}{2\pi} \int_0^{\infty} \left[e^{-\lambda z} + \Theta(\lambda)e^{-\lambda z} + X(\lambda)e^{+\lambda z}\right] j_0(\lambda r)d\lambda \ldots(3.9)$$

Où Θ(λ) et X(λ) sont des fonctions arbitraires de λ, et qui ne sont pas les mêmes pour chaque couche. Alors le potentiel est donné par :

$$V_i = \frac{\rho_1 I}{2\pi} \int_0^{\infty} \left[e^{-\lambda z} + \Theta_i(\lambda)e^{-\lambda z} + X_i\ (\lambda)e^{+\lambda z}\right] j_0(\lambda r)d\lambda \quad \ldots.(3.10)$$

Où l'indice i représente l'ordre de couche.

c. Adaptation de la solution aux conditions aux limites

Supposons les conditions aux limites suivantes :

- Le potentiel électrique est continu aux limites des différentes couches ;
- La composante verticale de la densité de courant électrique est continue aux limites des couches ;
- A la surface, la composante verticale de la densité de courant électrique ainsi que le champ électrique sont nuls ;
- Au voisinage de la source, le potentiel tend vers l'infini ;
- A des profondeurs très grandes, le potentiel tend vers zéro ;

Par la résolution d'un système d'équation, on obtient le potentiel à la surface écrit comme suit :

$$V = \frac{\rho_1 I}{2\pi}\int_0^{\infty}[1 + 2\Theta_1(\lambda)]j_0(\lambda r)d\lambda \ldots (3.11)$$

Où $\Theta_1(\lambda)$ est la fonction de Kernel de la première couche.

d. La fonction de Kernel

Posons $K(\lambda)= (1+2\Theta_1(\lambda))$, nous aurons :

$$V = \frac{\rho_1 I}{2\pi}\int_0^{\infty}[K_1(\lambda)]j_0(\lambda r)d\lambda \ldots (3.12)$$

La fonction $K(\lambda)$ a été introduite par Slichter (1933). A cet effet, on adopte la notation suivante :

$K(\lambda)$: la fonction Slichter-Kernel.

$\Theta_1(\lambda)$: la fonction Stefanescu-Kernel.

La fonction $K(\lambda)$ est liée aux paramètres de sous-sol par la relation récurrente de Pekeris donnée par :

$$K_i = \frac{[K_{i+1} + p_i \tanh(\lambda t_i)]}{[p_i + K_{i+1}\tanh(\lambda t_i)]} \quad \ldots.(3.13)$$

Où

$p_i = \rho_i/\rho_{i+1}$

$t_i=(h_i-h_{i-1})$

Koefoed (1970) a introduit la notion de la transformation de résistivité, notée Ti tel que :

$$T_i = \rho_i K_i$$

Alors la relation de Pekeris devient :

$$T_i = \frac{\left[T_{i+1} + \rho_i \tanh(\lambda t_i)\right]}{\left[1 + \frac{T_{i+1}\tanh(\lambda t_i)}{\rho_i}\right]} \text{....(3.14)}$$

e. La résistivité apparente

e.1 définition

Supposons la configuration illustrée dans la figure.(3.4). La différence de potentiel mesurée est donnée par :

$$\Delta V = 2(\rho I / 2\pi)\left[\frac{1}{s-b} - \frac{1}{s+b}\right] \text{....(1.15)}$$

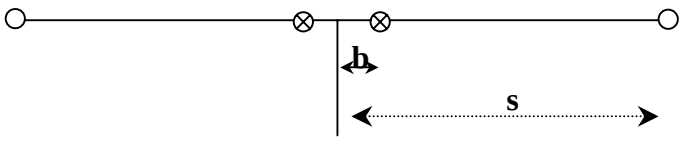


Figure 3.4 dispositif à configuration symétrique

Alors l'expression de la résistivité apparente est donnée par :

$$\rho_{app} = (\Delta V / I)2\pi s(s^2 - b^2)/(4bs) \text{...(3.16)}$$

Le terme $s.(s^2-b^2)/(4bs)$ est appelé facteur géométrique, et dépend de la configuration utilisée.

e.2. La relation entre la résistivité apparente et la transformation de résistivité

Dans l'équation (3.16), remplaçons ΔV par :

$$\Delta v = 2\left[V(s-b) - v(s+b)\right]$$

Tel que :

$$V = \frac{\rho_1 I}{2\pi}\int_0^\infty \left[K_1(\lambda)\right] j_0(\lambda r) d\lambda$$

Alors nous obtenons :

$$\rho_{app} = 2s\frac{1-c^2}{4c}\int_0^{\infty} T(\lambda)\{j_0[\lambda s(1-c)] - j_0[\lambda s(1+c)]\}d\lambda \quad \cdots (3.17)$$

Où c=b/s est l'excentricité.

- Si le dispositif est de Wenner, alors :

$$\rho_{appW} = 2a\int_0^{\infty} T(\lambda)[j_0(\lambda a) - j_0(2\lambda a)]d\lambda \quad \ldots. (3.18)$$

- Si le dispositif est de Schlumberger, alors :

$$\rho_{app.Schl.} = -\frac{2\pi s^2}{I}(\partial V/\partial r)_{r=s} \quad \ldots. (3.19)$$

$(s^2-b^2)/(4bs) \longrightarrow s/4b$
$\Delta V/2b \longrightarrow 2(\partial V/\partial r)_{r=s}$
Substituons dans l'equation (3.16) on aura :

$$\frac{\partial}{\partial x} j_0(x) = -j_1(x) \quad \ldots. (3.20)$$

Utilisons la relation suivante :

$$\rho_{app.Schl} = \rho_1 s^2 \int_0^{\infty} K(\lambda) j_1(\lambda s) d\lambda \ldots (3.21)$$

Nous obtenons :

$$V = \frac{\rho_1 I}{2\pi r} + 2\int_0^{\infty} \Theta_1(\lambda) j_0(\lambda r) d\lambda \quad \ldots. (3.22)$$

Cet intégral diverge, pour lever cette indétermination, on utilise l'expression de **p**otentiel en fonction de $\Theta(\lambda)$ (fonction Stefanescu-Kernel), on aura :

$$\rho_{app.Schl.} = \rho_1 + s^2 \int_0^{\infty} [T(\lambda) - \rho_1] j_1(\lambda s) \lambda d\lambda \ldots (3.23)$$

III.3. CALCUL DE LA COURBE DE RESISTIVITE APPARENTE

Supposons qu'on a le modèle ci-dessous :

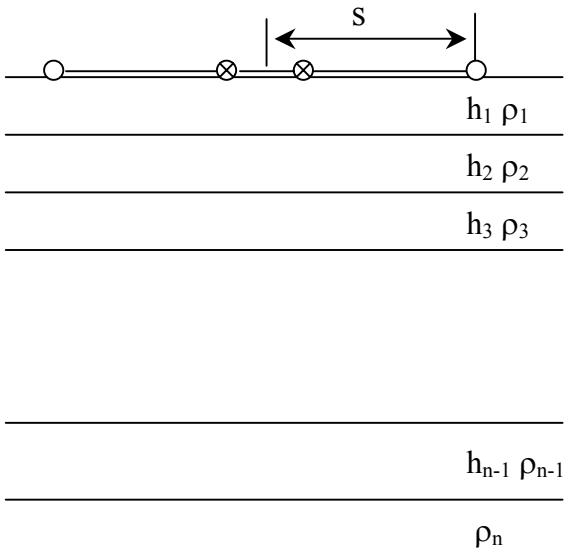


Figure3.5 Modèle simple de sous-sol

Il existe plusieurs méthodes de calcul de la courbe de résistivité apparente. On trouve principalement :
- Calcul numérique d'intégral ;
- Calcul par la méthode point image ;
- Calcul par décomposition en fraction partielle ;
- Calcul par la méthode de filtre linéaire.

Dans cette partie, on se contente de décrire le principe de la première méthode.

La résistivité apparente est donnée par :

$$\rho_{app.Schl.} = \rho_1 + s^2 \int_0^{\infty} [T(\lambda) - \rho_1] j_1(\lambda s) \lambda d\lambda \quad \ldots(3.24)$$

On fait un échantillonnage de cette relation, on obtient :

$$\rho_{app}(s) = \rho_1 + s^2 \sum_{k=0}^{M} [T_1(kd\lambda) - \rho_1] j_1(kd\lambda s) kd\lambda^2 \ldots(3.25)$$

Les valeurs de $T_1(kd\lambda)$ sont calculées à partir de la relation récurrente suivante :

$$T_i = \frac{T_{i+1} + \rho_i \tanh(kd\lambda t_i)}{1 + \frac{T_{i+1} \tanh(kd\lambda t_i)}{\rho_i}} \ldots..(3.26)$$

Tel que :
t_i :l'épaisseur de la I^{ème} couche.
$T_n=\rho_n$
J_1 : fonction de Bessel d'ordre 1.

III.4. INVERSION PAR RNA

III.4.1 Position de problème

Le problème consiste à trouver les poids du réseau de neurones qui permet de prédire le modèle électrique de sous-sol à partir des data mesurés.

La résolution de ce problème nécessite deux étapes fondamentales

- Etape d'apprentissage ;
- Etape de généralisation.

III.4.2 Etape d'apprentissage

Dans cette étape, plusieurs exemples synthétiques sont utilisés et choisis au hasard avec quelques spécifications. Ceux-ci permettent de tracer des cartes souvent non-linéaires entre les data entrants et sortants, et ce afin de tirer le maximum d'informations sur le problème à étudier.
L'étape d'apprentissage est le processus qui permet de calculer les poids par minimisation de la différence entre les data sortants et les data désirés.
Il est important de normaliser les données d'apprentissage, car la fonction d'activation est limitée entre 0 et 1 (Jimmy t al., 2004).
Supposons que nous avons les vecteurs synthétiques entrants-sortants, notés par :

$$(\overset{1}{x},\overset{1}{y}),(\overset{2}{x},\overset{2}{y}),.....(\overset{q}{x},\overset{q}{y}),.....(\overset{Q}{x},\overset{Q}{y})$$

Tel que :

$$x=(\rho_{a1},\rho_{a2},\rho_{a3},.....,\rho_{aN})$$

$$y=(h_1,h_2,.....h_{m-1},\rho_1,\rho_2,.....,\rho_m)$$

x vecteur entrant à N éléments
y vecteur sortant à M=2m-1 éléments

Supposons un réseau de neurones à deux couches, le vecteur $\overset{q}{x}$ se propage dans la couche cachée selon l'équation suivante (Carlos et et al., 2000) :

$$a_j=\overset{h}{\varphi}(\sum_{i=1}^{\overset{h}{N}}\overset{h}{W}_{ji}\overset{q}{x}_i+b_j)\cdots.(3.27)$$

Avec J=1,L (L nombre de neurones dans la couche cachée).

Les outputs sont donnés par la relation suivante (Carlos et al., 2000) :

$$o_k^q = f^0\left(\sum_{j=1}^{L}(W_{kj}\, a_j^q + c_k)\right) \ldots (3.28)$$

Et k=1,M le nombre de neurones dans la couche output.

L'erreur est donnée par la relation suivante (Carlos et al., 2000) :

$$E = \frac{1}{2}\sum_{q=1}^{Q}\sum_{k=1}^{M}(y_k^q - o_k^q)^2 \ldots (3.29)$$

L'apprentissage est réalisé par l'algorithme de rétro-propagation.

III.5 INVERSION PAR RECUIT SIMULE

III.5.1 Introduction

Recuit simulé « *Simulated Annealing* » est une technique stochastique combinatoire basée sur le mécanisme statistique, la thermodynamique et la probabilité à plusieurs variables. Elle est également une technique itérative de Monte Carlo dans laquelle le processus d'optimisation d'une fonction objectif est remplacé par un processus de « *Cooling* » et « *Annealing* » (Dittmer et Szymanski, 1995).

Cette technique a été utilisée dans la résolution de plusieurs problèmes physiques. Metropolis et al. (1953) l'ont utilisée pour résoudre les équations d'états de sub-surface. Bonomi et Lutton (1984) l'ont utilisée pour résoudre le problème dit «*traveling salsman problem* ». Elle a été également utilisée dans le traitement d'image.

Dans la géophysique, recuit simulé a été appliquée dans le calcul des corrections statiques et dans l'inversion sismique (Sen et Stoffa 1991)

III.5.2 Concept de base

Considérons un sub-surface dans un état liquide et chaud. Les molécules possèdent une quantité importante d'énergie et par conséquent une large quantité de mobilité thermique, donc ils bougent rapidement et d'une manière chaotique.
Lorsque le liquide se refroidit, l'énergie est graduellement perdue et la solidification et la cristallisation commencent. Les molécules deviennent moins mobiles et s'installent dans une position qui décroît l'énergie du système.
Si le refroidissement s'est fait avec un taux normal, l'énergie du système sera minimale, mais si le refroidissement s'est effectué d'une manière rapide l'état d'énergie du système sera élevé.

L'énergie du système est directement liée à la température et la position des molécules. Alors si le refroidissement s'est effectué avec un taux suffisant, le processus d'*Annealing* empêche naturellement le système de se piéger dans un état d'énergie élevée.

Un système peut prendre un nombre fini n de configurations, chacune d'elles a la probabilité π_i.

Initialement, une configuration χ_l est choisie. Un changement aléatoire à χ_l donne une nouvelle configuration χ_{l+1} .

$$P(\Delta E) = \exp(-\Delta E / kT) \quad \ldots(3.30)$$

Ceci est l'algorithme de Metropolis qui représente un élément primordial dans la technique « S*imulated Annealing ».*
Deux cas peuvent se présenter
1- $\Delta E \leq 0$: la probabilité de 1 est assignée, ce changement est toujours accepté.
2- $\Delta E > 0$: la probabilité est calculée par l'utilisation de l'algorithme de Metropolis. Dans la pratique, ce changement est accepté si un nombre généré aléatoirement entre 0 et 1 est inférieur à la valeur déterminée par l'algorithme de Metropolis.

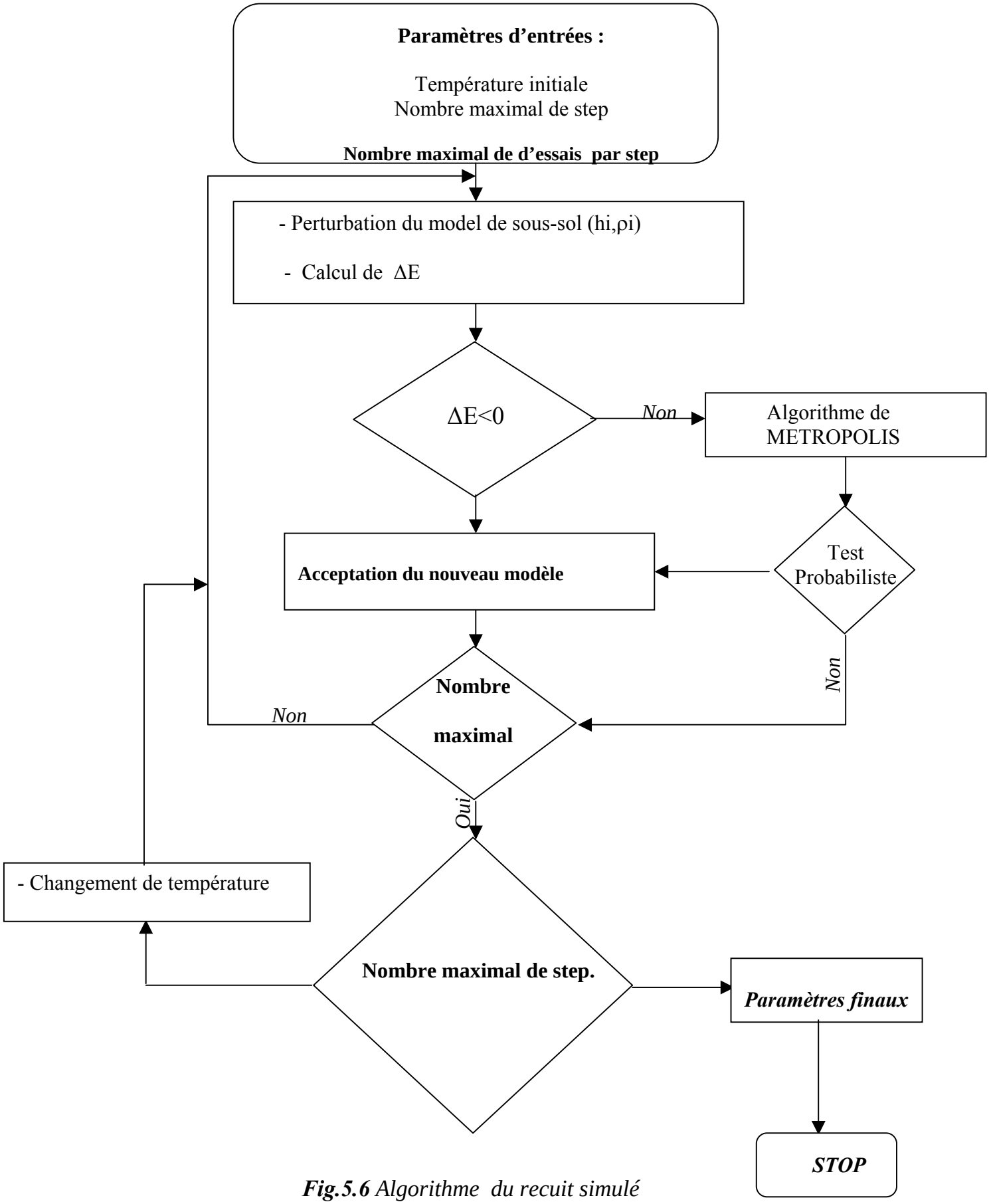


Fig.5.6 *Algorithme du recuit simulé*

Chapitre IV

IV.1. APPLICATIONS SISMIQUES.

IV.1.1 Pointé des horizons sismiques par réseaux de neurones artificiels

a. Cas synthétiques

• Dans la figure (4.1), nous avons pris un signal Ricker avec une longueur de 17 échantillons et une fréquence centrale fo de 25Hz.

Ce Ricker est considéré comme un signal de référence qui sera utilisé durant le processus d'apprentissage. Autrement dit, notre réseau est en train d'apprendre la capacité de détection et de reconnaissance du Ricker pendant cette étape d'apprentissage.

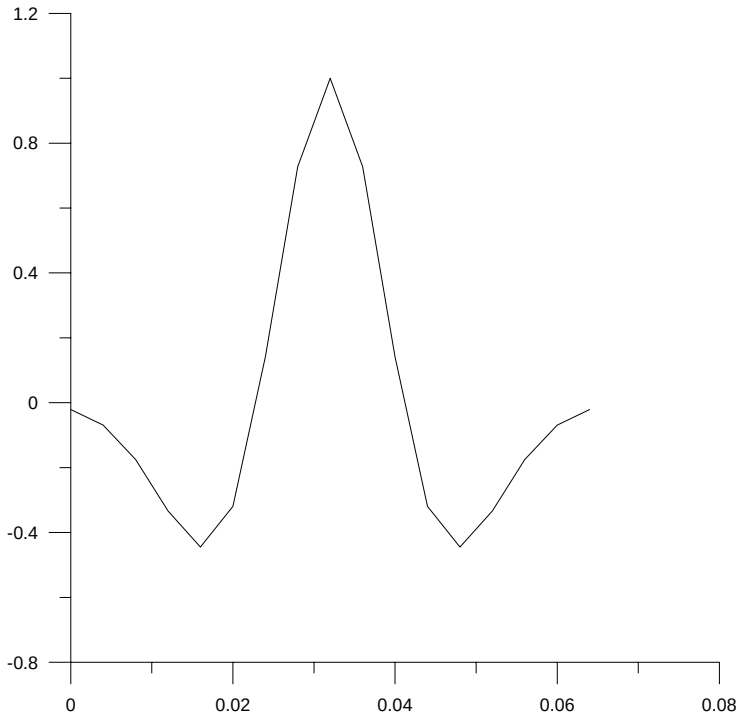


Fig.4.1 signal référence (Ricker)

• Dans la figure (4.2), nous présentons la topologie du réseau de neurones utilisé pour la détection de forme. Il est constitué de deux couches, une couche cachée contenant quatre neurones et une couche de sortie formée d'un seul neurone.

Nous avons pris six inputs qui sont l'espérance mathématique de la transformé d'Hilbert, de l'amplitude instantanée, de la phase instantanée, de l'énergie sur la fenêtre, de l'entropie et du signal réel. Par ailleurs, nous avons choisi à l'output une seule valeur égale à l'unité.

Le tableau (4.1) montre les attributs sismiques du signal de référence.

Les tableaux (4.2.a) et (4.2.b) montrent les valeurs des poids du réseau, calculées par le logiciel Pythia.

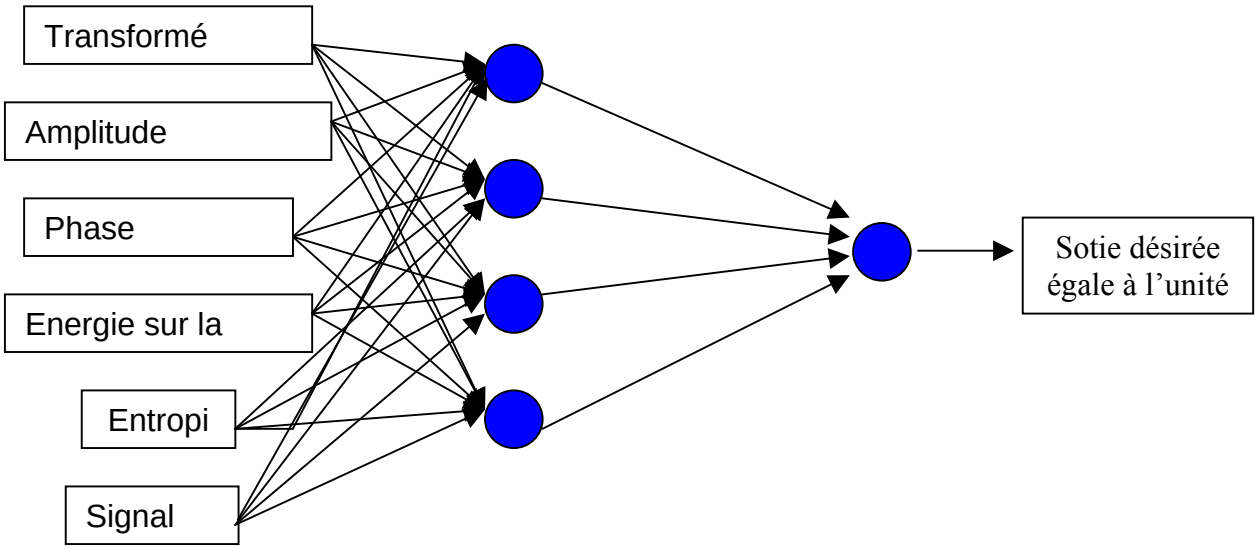


Fig.4.2. Réseau d'apprentissage.

Transformé d'Hilbert	-0,023384
Amplitude instantanée	0,173227
Phase instantanée	0,687223
Energie sur la fenêtre	2,992015
Entropie	0,038565
Signal réel	0,000774

Table N°4.1. Attributs sismiques du Ricker.

LA COUCHE CACHEE			
Neurone 1	***Neurone 2***	***Neurone 3***	***Neurone 4***
-0.434597	0.483725	0.1403	0.529062
-0.390684	0.482077	-0.735649	0.736154
0.203937	-0.716909	-0.272435	-0.698237
0.3165	0.836865	0.905425	-0.271708
-0.602593	0.361453	0.838516	0.092131
-0.003595	-0.675967	-0.102733	0.761129

Table (4.2.a). Poids de la couche cachée.

LA COUCHE OUTPUT
Neurone 1
-2.273235
-2.019278
-2.330592
-1.137888

Table (4.2.b). Poids de la couche output.

• Dans la figure (4.3), nous avons une trace synthétique de longueur 500 échantillons à trois pics dont les positions sont : 80 échantillons, 180 échantillons et 237 échantillons. Nous avons appliqué le réseau de neurones calculé pour la détection de ces trois pics. La méthodologie consiste à calculer le long d'une fenêtre glissante les attributs sismiques et les injecter à l'input du réseau. A l'output, le pic correspond à la valeur la plus proche à l'unité.
Pour la fenêtre glissante, nous avons utilisé les quatre longueurs suivantes : 12, 16, 20, 24 échantillons. Les résultats correspondants sont présentés dans les figures (4.4), (4.5), (4.6) et (4.7).

Dans le cas d'une longueur de la fenêtre glissante de 12, 16 et 24 échantillons, nous constatons que les trois pics ont été bien repérés. Par contre, ils n'ont pas pu être détectés dans le cas restant, c'est-à-dire pour une longueur de 20 échantillons, ceci montre l'importance cruciale du choix de la longueur de la fenêtre glissante dans l'application de cette technique.

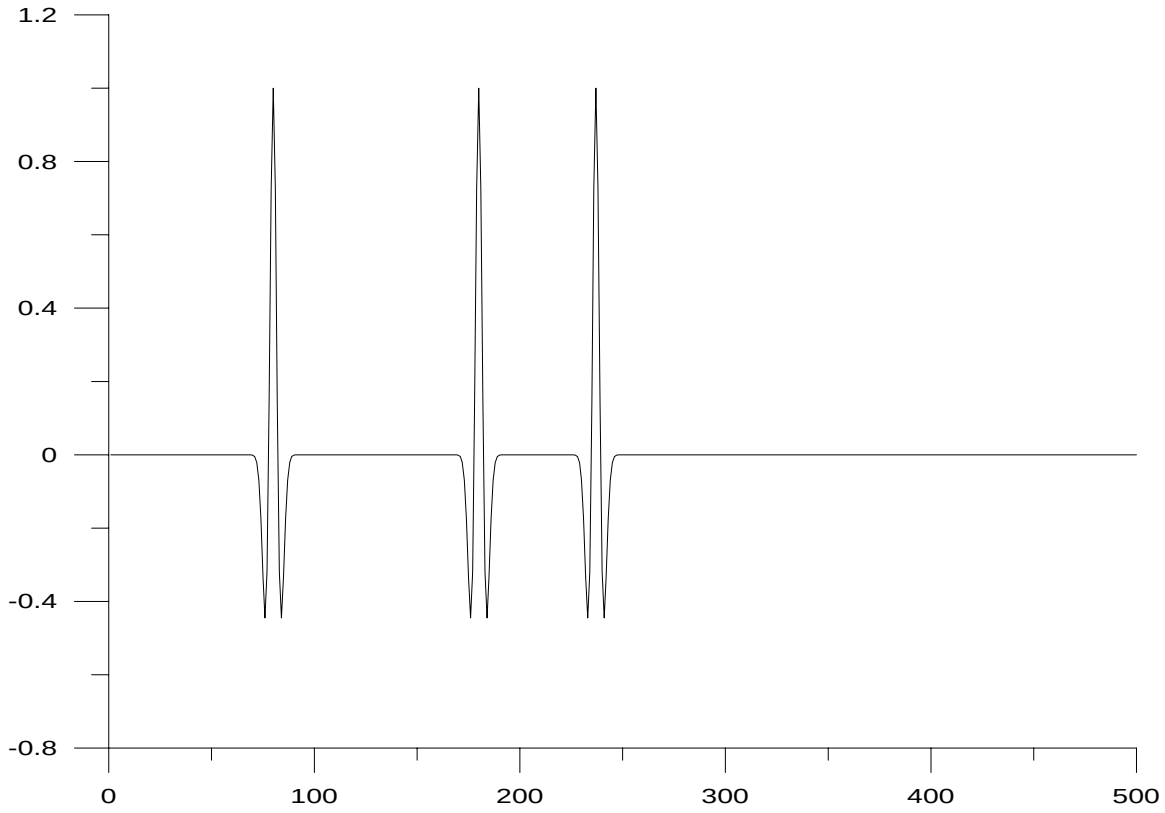


Fig.4.3. Trace synthétique.

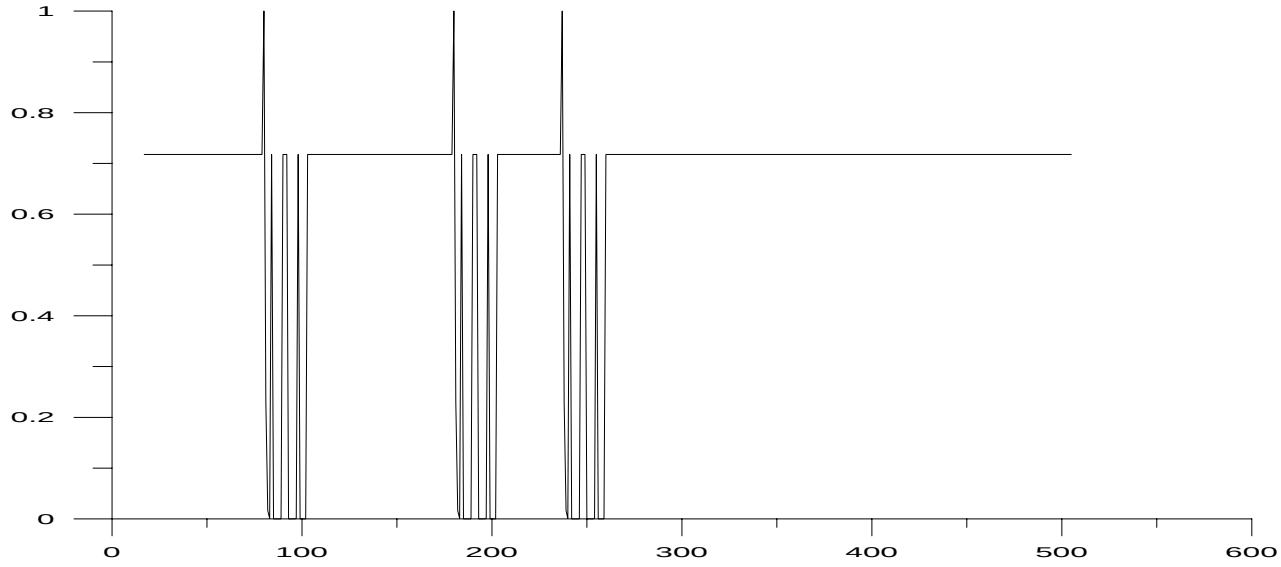


Fig.4.4 Output (fenêtre glissante de 12 échantillons)

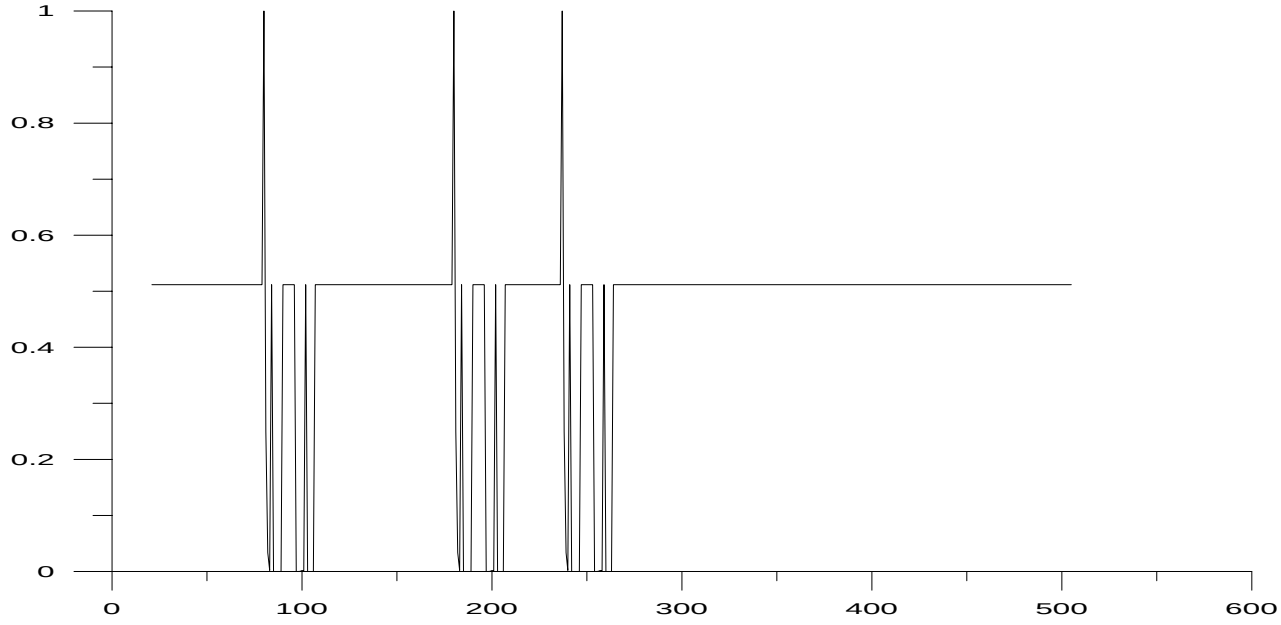


Fig.4.5 Output (fenêtre glissante de 16 échantillons)

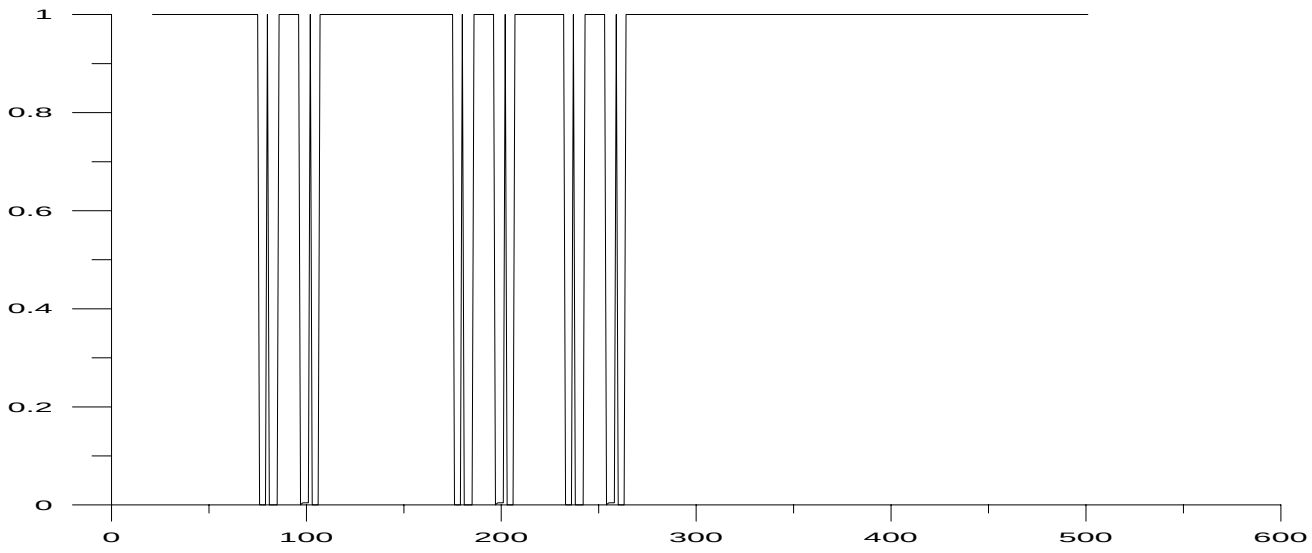


Fig.4.6 Output (fenêtre glissante de 20 échantillons)

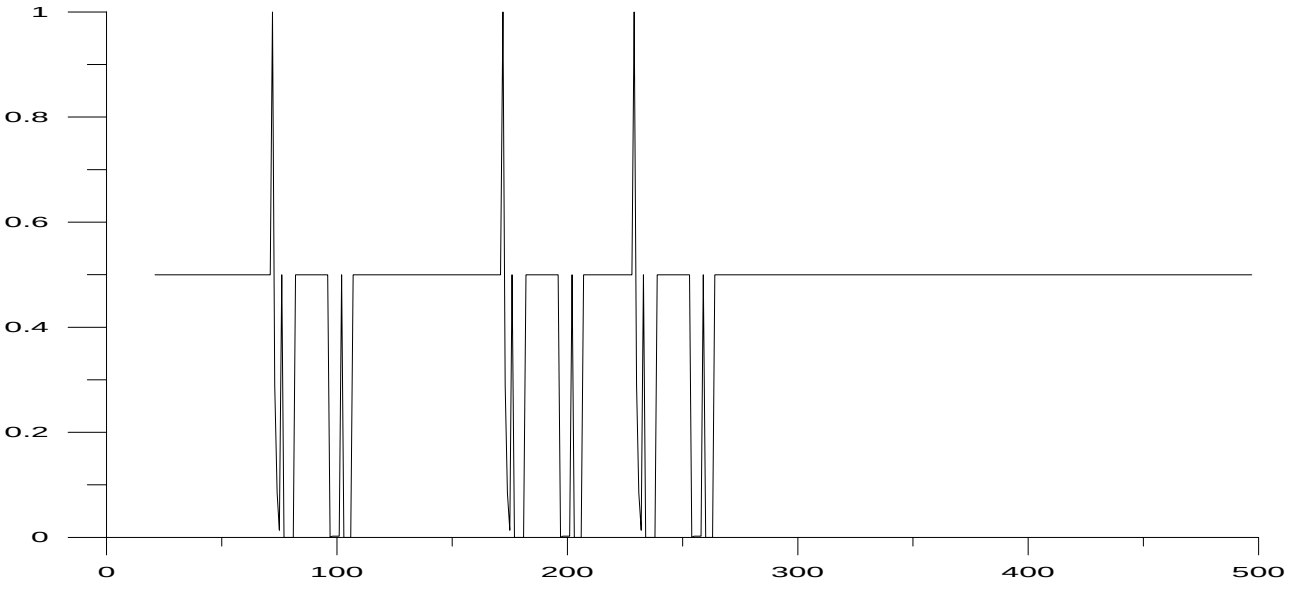


Fig.4.7 Output (fenêtre glissante de 24 échantillons)

• Dans les figures (4.8), (4.10) et (4.12), nous avons essayé cette technique pour des traces bruitées avec un taux respectivement égal à 5%, 30% et 50%. Ce taux du bruit mesure le degré de contamination du signal par du bruit, il s'exprime sous la forme du rapport des puissances respectives du signal et du bruit (Coulon, 1996). Les résultats obtenus sont présentés dans les figures (4.9), (4.11) et (4.13). Nous constatons la sensibilité de la performance de cette technique au taux du bruit utilisé. Alors, pour un taux de 5% la technique n'a pas pu détecter le dernier pic et elle a échoué également à repérer les deux derniers pics dans le cas d'un taux égal à 30% et 50%.

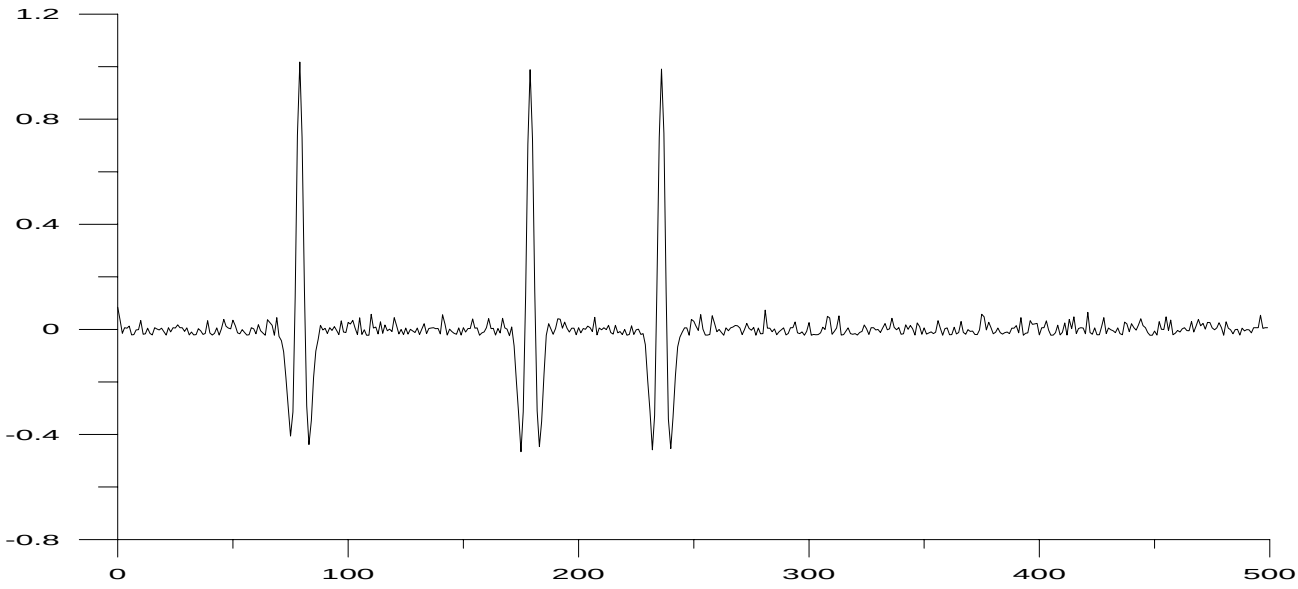


Fig.4.8 Trace bruitée (taux de bruit =5%)

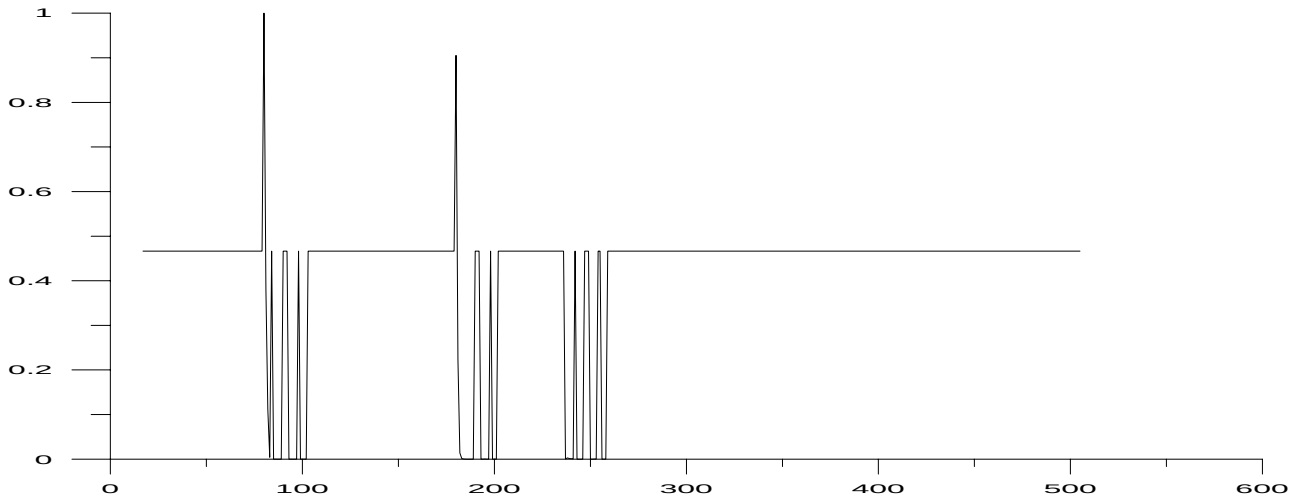


Fig.4.9 Output (taux de bruit =5%)

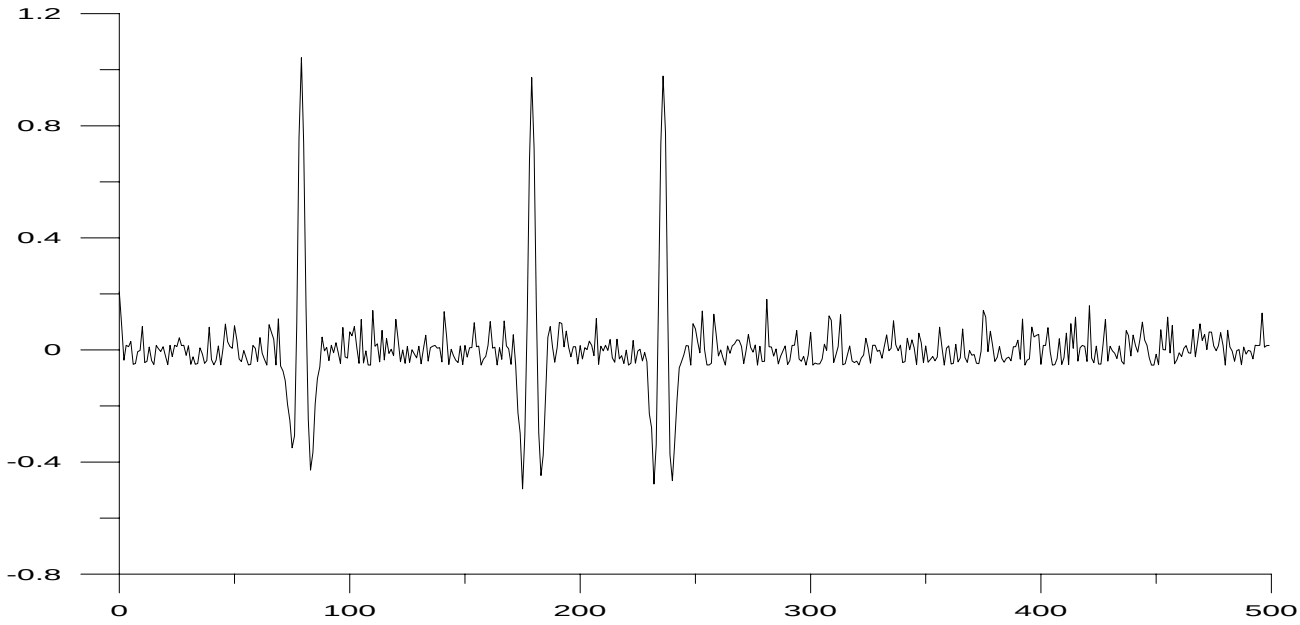


Fig.4.10 Trace bruitée (taux de bruit =30%)

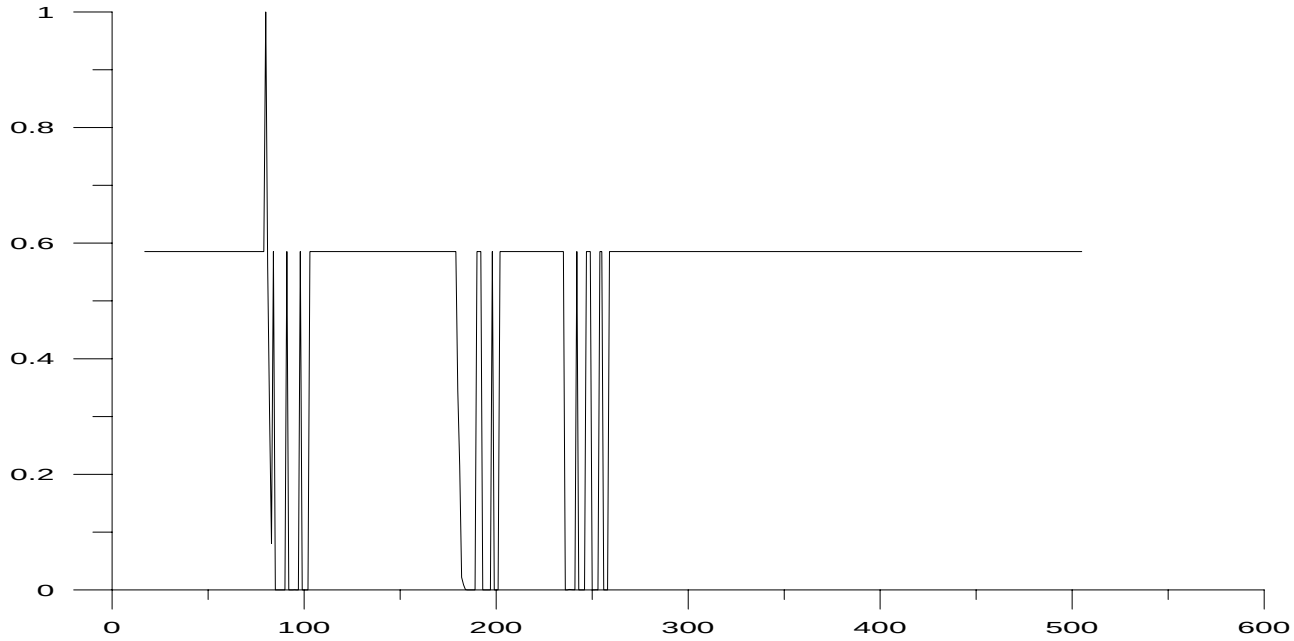


Fig.4.11 Output (taux de bruit =30%)

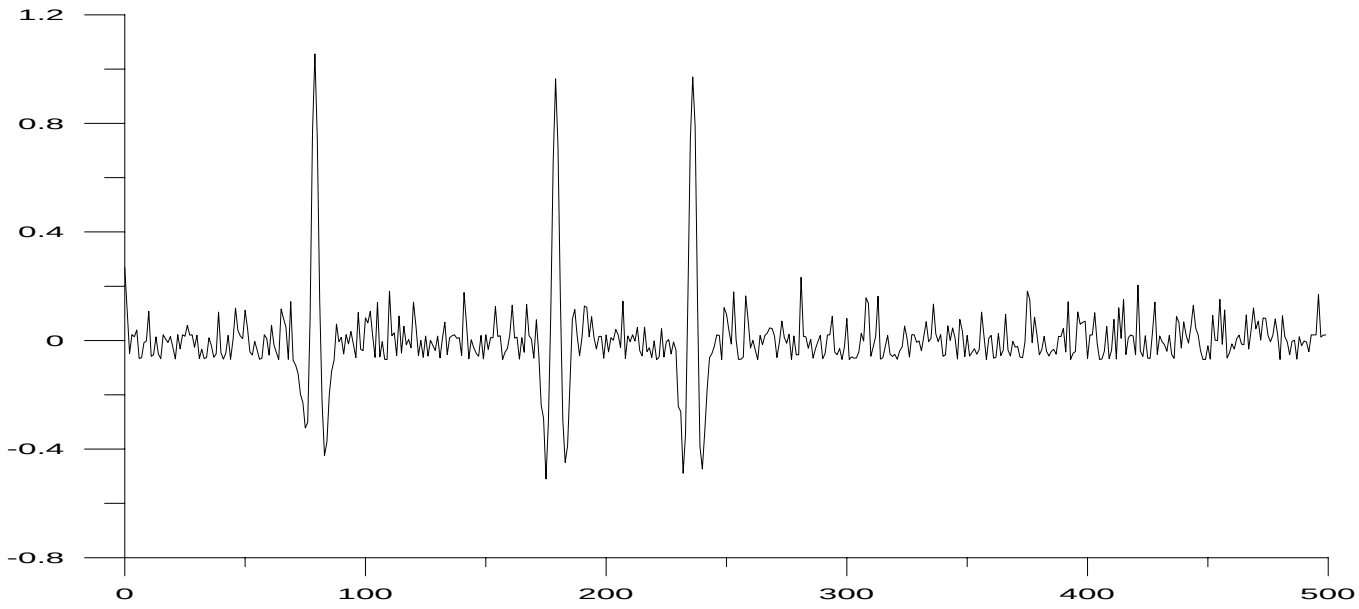


Fig.4.12 Trace bruitée (taux de bruit =50%)

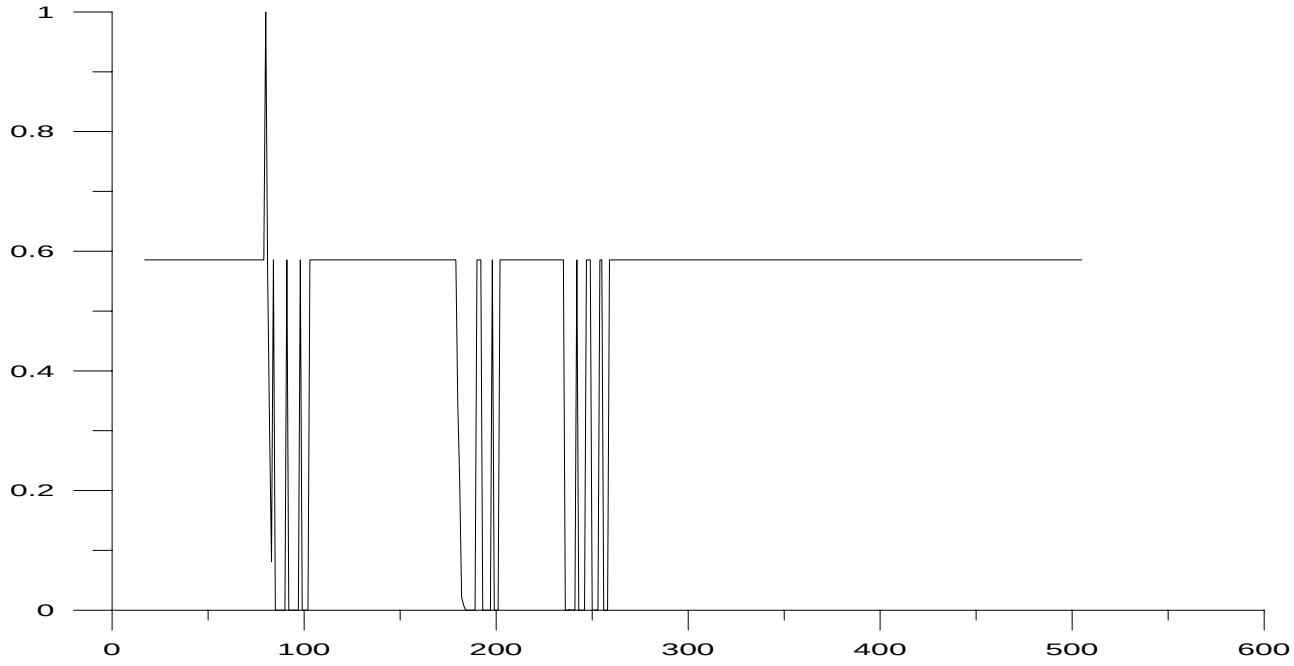


Fig.4.13 Output (taux de bruit =50%)

Nous avons appliqué cette technique à un film synthétique à trois couches dont les paramètres sont les suivants :

H1=100m, H2=300m, H3=400 m ;
V1=1000m/s V2=1500m/s, V3=3500m/s V4=4000 m/s ;
Nombre de trace = 20 ;
Nombre d'échantillon =500 ;
Taux d'échantillonnage =0.004s.

Dans la figure (4.14), nous avons présenté le film à trois couches. Après avoir appliqué une fenêtre glissante d'une longueur variante d'une trace à l'autre d'une manière qu'elle puisse détecter les réflecteurs, nous avons présenté le résultat de l'Output de notre réseau sur la figure (4.15). Nous remarquons que le dernier réflecteur n'a pas pu être détecté par le réseau de neurones.
Comme la longueur de la fenêtre est un paramètre très déterminant, nous devons appliquer à chaque trace sa propre fenêtre déterminée par test.

Dans les figures (4.16), (4.17), (4.18) et (4.19), nous avons appliqué cette technique à des films synthétiques bruités avec des taux de 5% et 50%. Nous constatons que les deux derniers réflecteurs sont indétectables par cette technique ; ce qui confirme la sensibilité de cette méthode à la qualité du data.

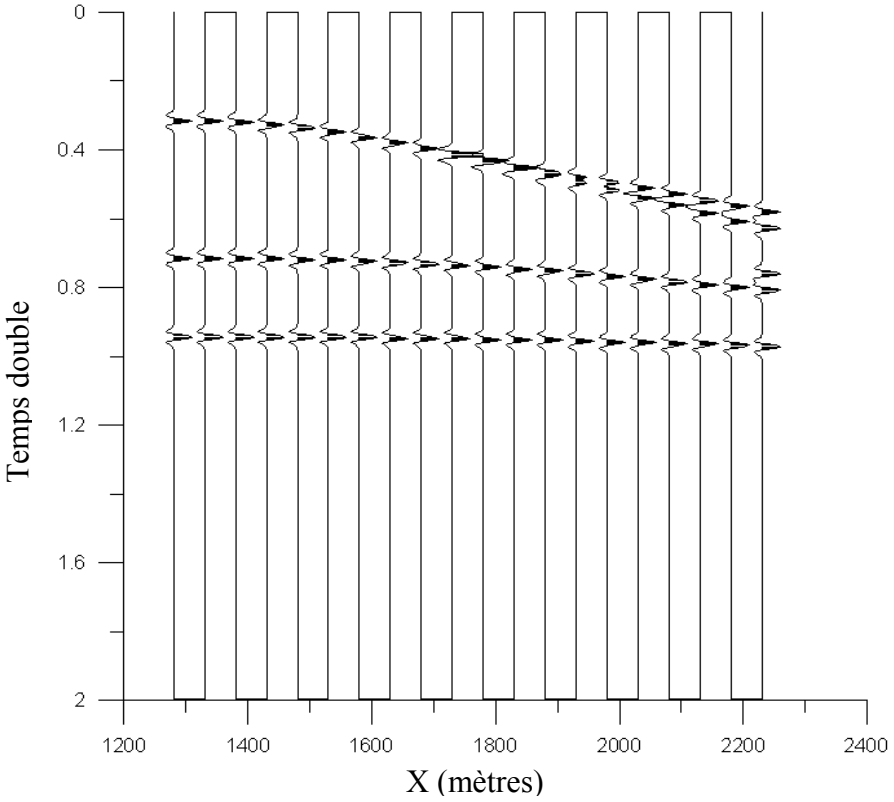


Fig.4.14 Film synthétique

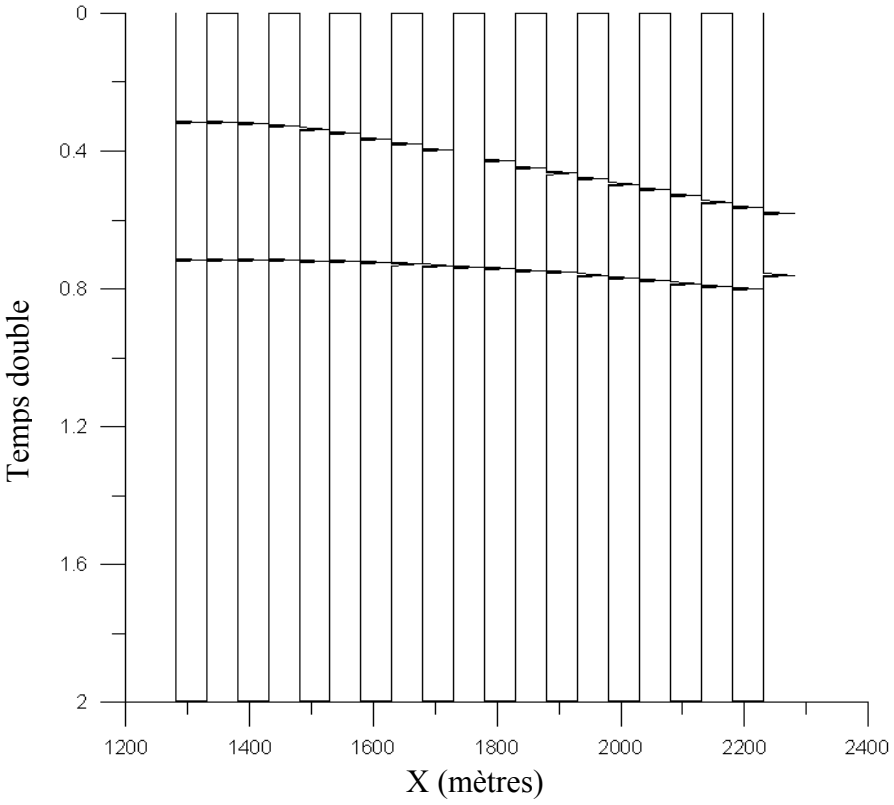


Fig.4.15 Output

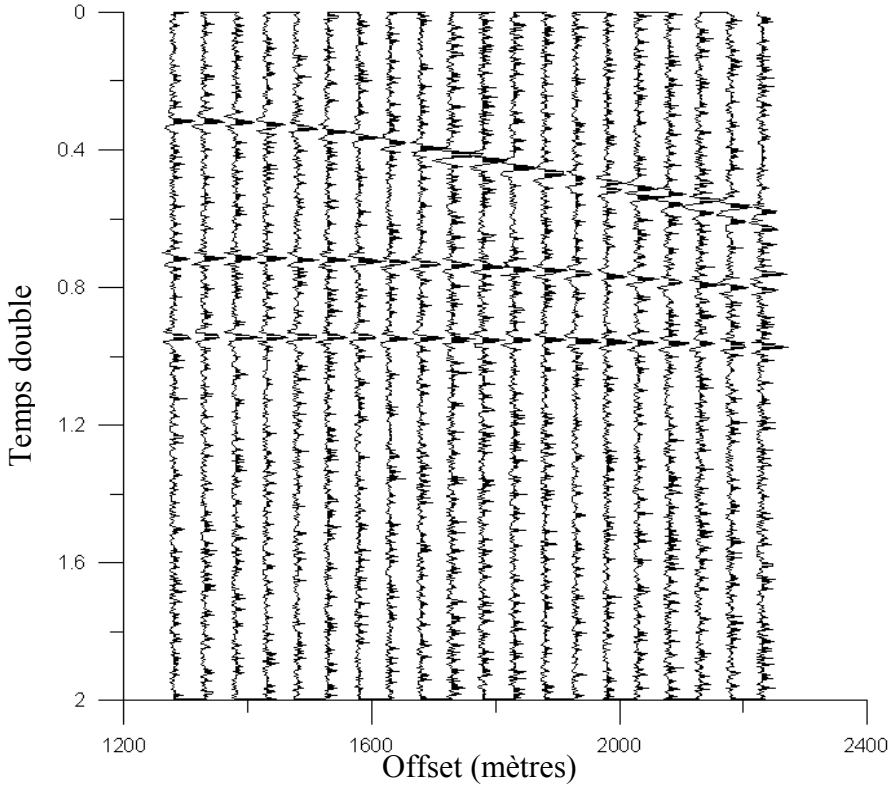


Fig.4.16 Film synthétique (Taux de bruit 5%)

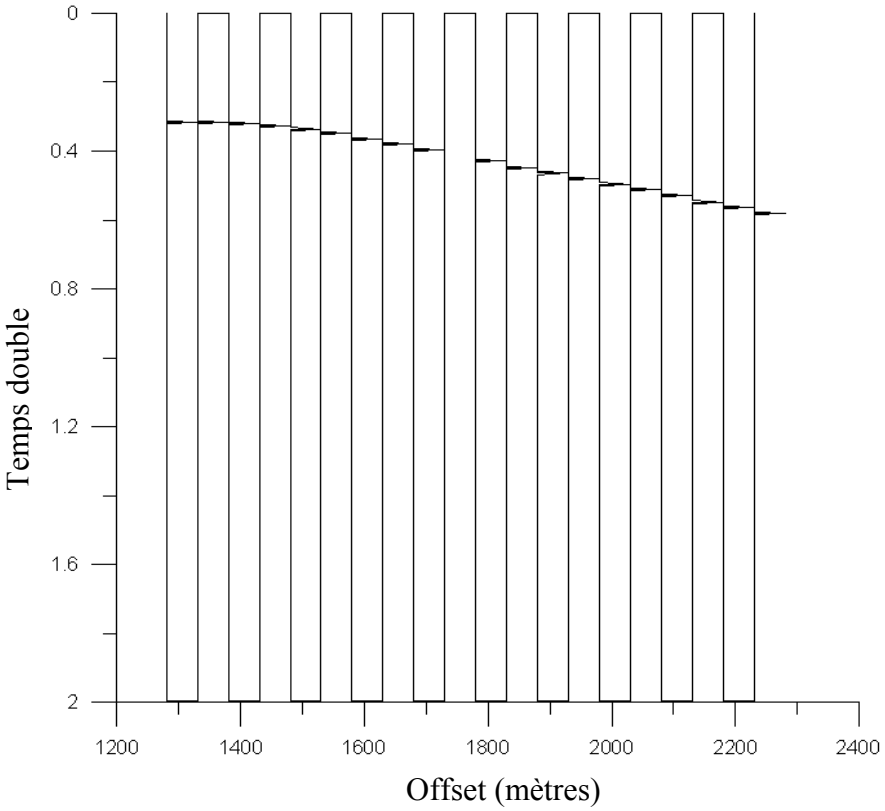


Fig.4.17 Output (Taux de bruit 5%)

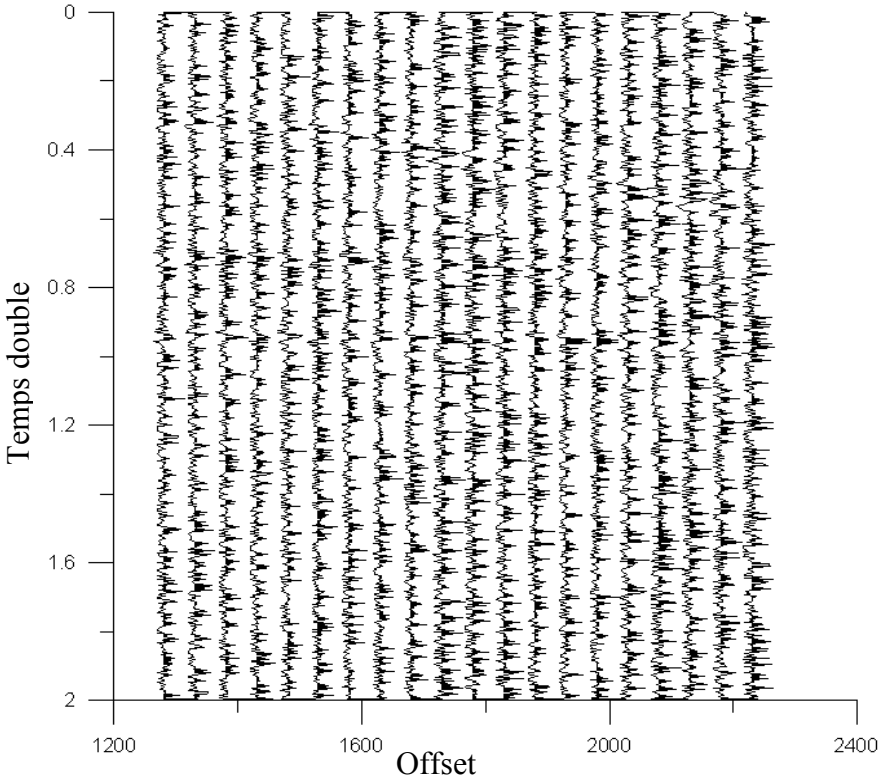


Fig.4.18 Film synthétique (Taux de bruit 50%)

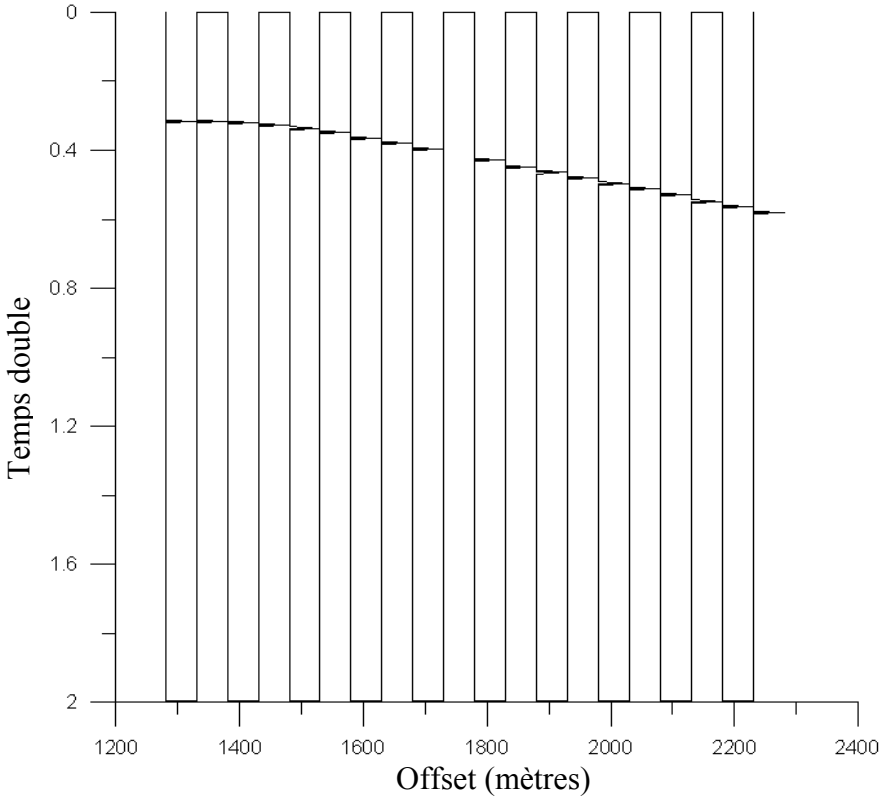


Fig.4.19 Output (Taux de bruit 50%)

b. Cas réel :

• Dans la figure (4.20), nous représentons une portion d'une section sismique réelle ayant 100 CDP dont les paramètres sont :
Temps d'enregistrement=5secondes ;
Taux d'échantillonnage = 2ms, taux de rééchantillonnage= 4ms ;
Couverture multiple =140, l'intertrace =40m, l'Offset = 60m
Ces données ont été enregistrées dans la région EL ERF, Gassi Touil, Rhourde Nouss.

A fin de pointer l'horizon indiqué par une ligne dans la section sismique, pour cela nous avons pris une forme de référence définie par les paramètres suivants :
- Position : la première trace entre l'échantillon 434 et 443 ;
- Longueur = 10 échantillons.
Cette forme est présentée dans le figure (4.21) et dont les attributs sismiques sont présentés dans le tableau N°4.3. La topologie du réseau utilisé est présentée dans la figure (4.22), il s'agit d'un réseau à deux couches dont la couche cachée est constituée de quatre neurones et la couche output est formée d'un seul neurone.

• Apres avoir effectué le processus d'apprentissage par le logiciel Pythia, nous avons présenté les différents poids synaptiques du réseau obtenu sur les tableaux (4.4) et (4.5). Nous avons appliqué ce réseau à la première trace de la section sismique présentée dans la figure (4.23). Après plusieurs essais de la longueur de la fenêtre, nous avons obtenu une longueur de 6 échantillons donnant un pic qui correspond bien à l'objectif défini initialement

L'application de cette fenêtre à la portion de la section sismique donne la section output présentée dans la figure (4.25). Ce résultat est médiocre puisque la forme de référence n'a pas pu être détectée le long de tout les 100 CDP.

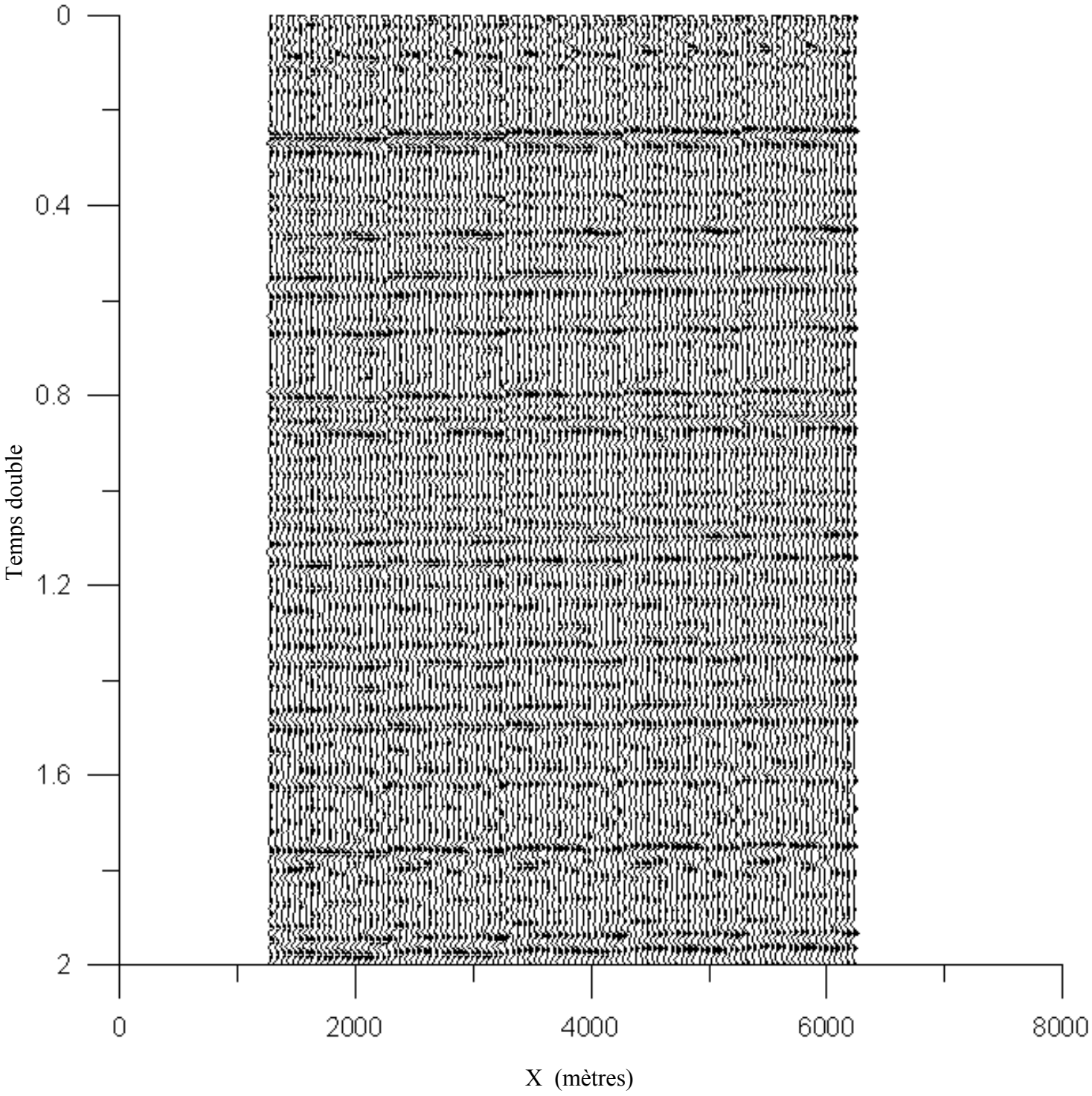


Fig.4.20.Section réelle.

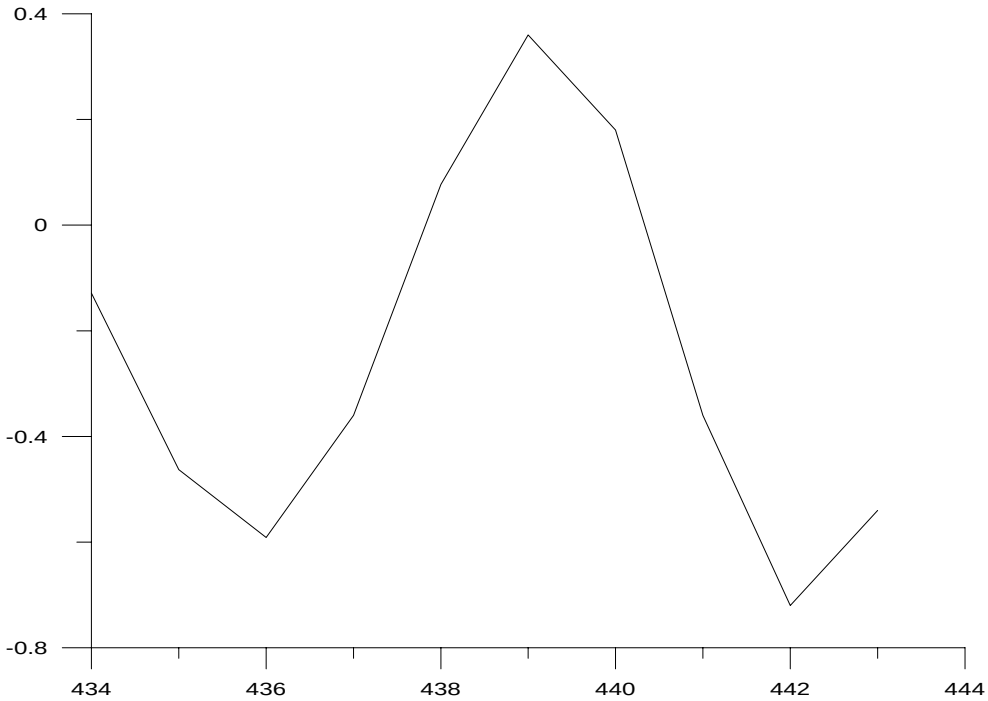


Figure 4.21. La forme de référence.

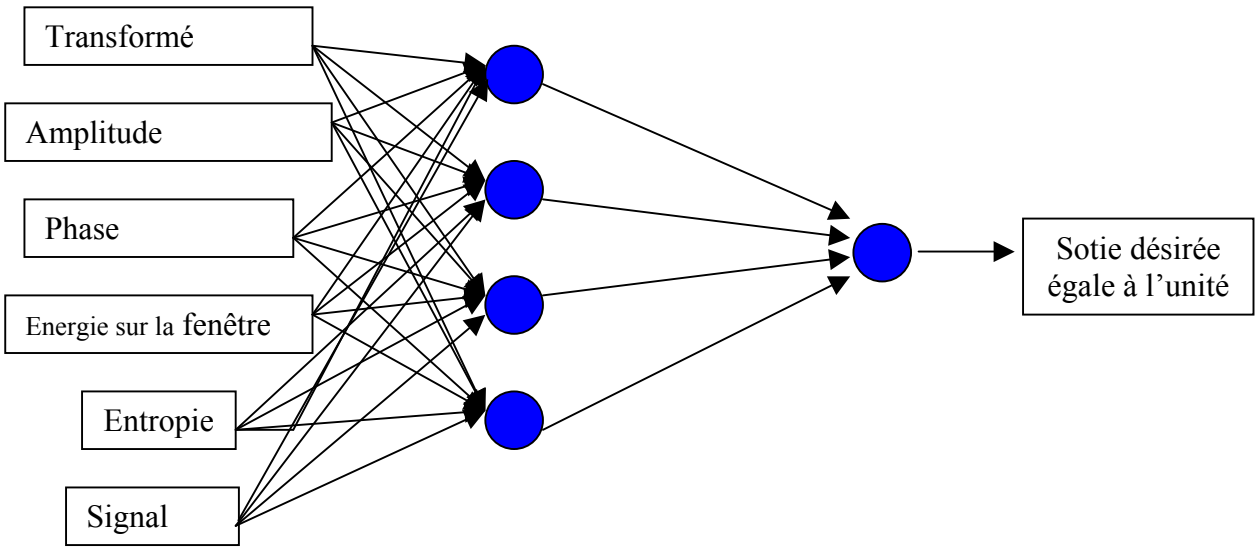


Figure 4.22. Réseau de neurones.

LA COUCHE CACHEE			
Neurone 1	**Neurone 2**	**Neurone 3**	**Neurone 4**
-0.372401	0.499466	0.546841	-0.385457
-0.343870	-0.467551	-0.940895	-0.949160
-0.447253	0.813185	-0.20915	-0.674730
0.811968	0.978115	0.066356	0.164580
-0.479494	-0.980338	-0.152995	0.209135
-0.715954	-0.134907	-.129019	0.732974

Table N°4.4. Poids de la couche cachée.

LA COUCHE OUTPUT
Neurone 1
-1.1914762
-1.467953
-2.312721
-2.783663

Table N°4.5. Poids de la couche output.

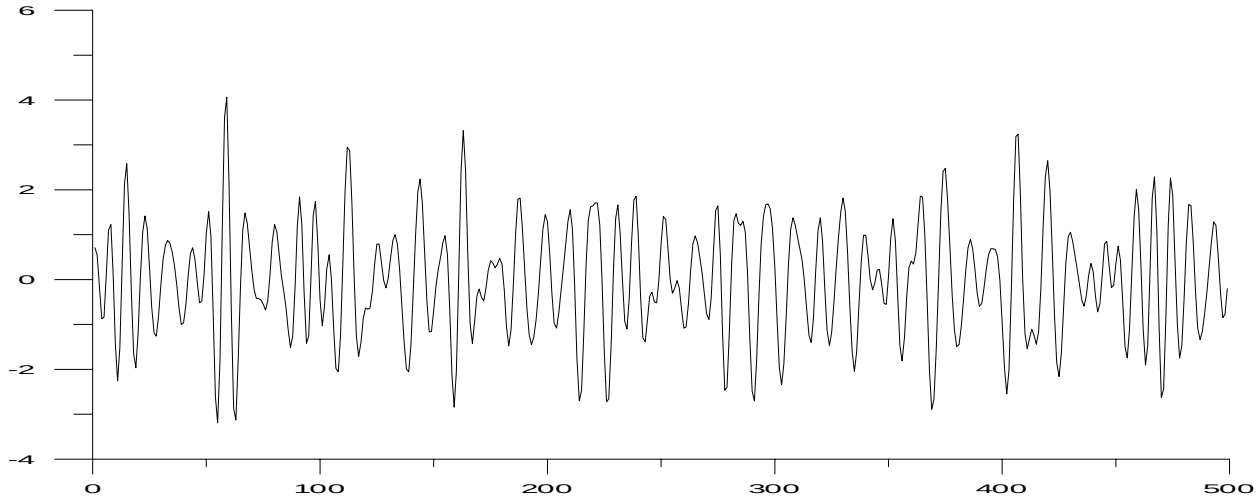


Fig.4.23. La première trace de la section réelle.

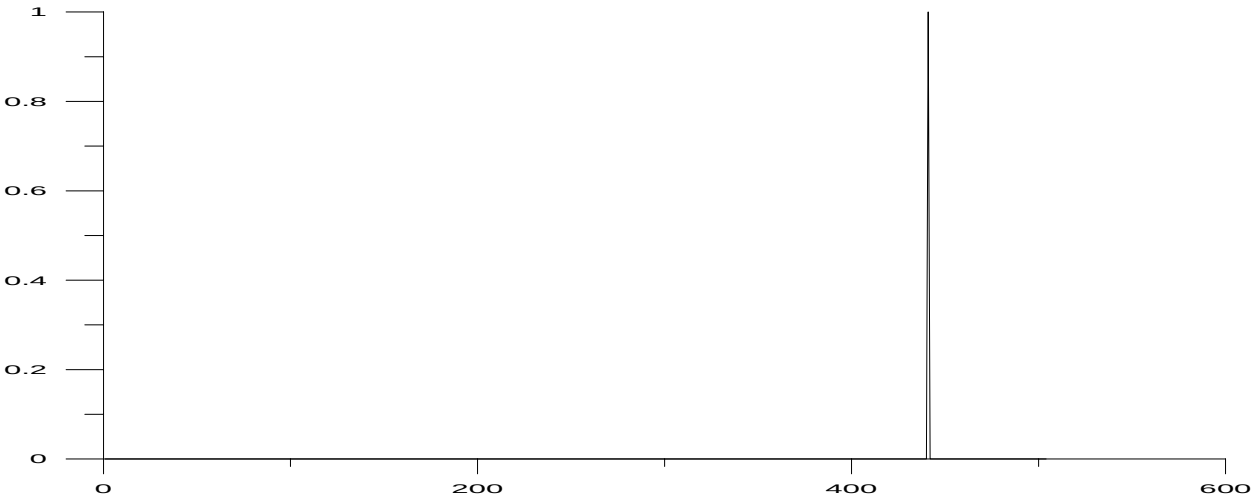


Fig.4.24. Output de la première trace.

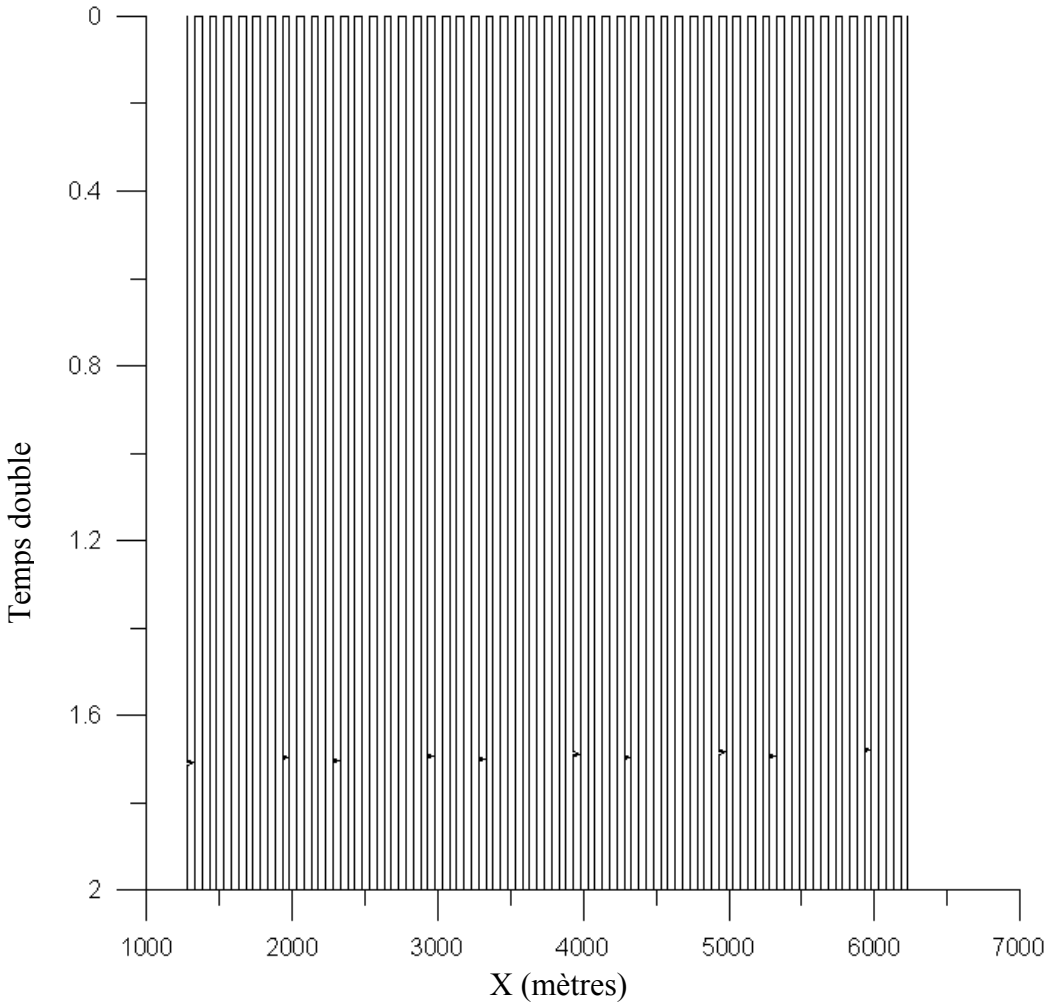


Fig.4.25. Output de la section réelle.

IV.1.2 Pointé des horizons sismiques par moments invariants

a. Cas synthétiques

- Dans la figure (4.26), il est présenté la forme de référence qui est un signal Ricker de longueur 17 échantillons et de fréquence centrale 25Hz. Nous avons calculé les différents ordres des moments de cette forme référence, nous avons obtenu la matrice dite moment invariant présentée dans la figure (4.27).

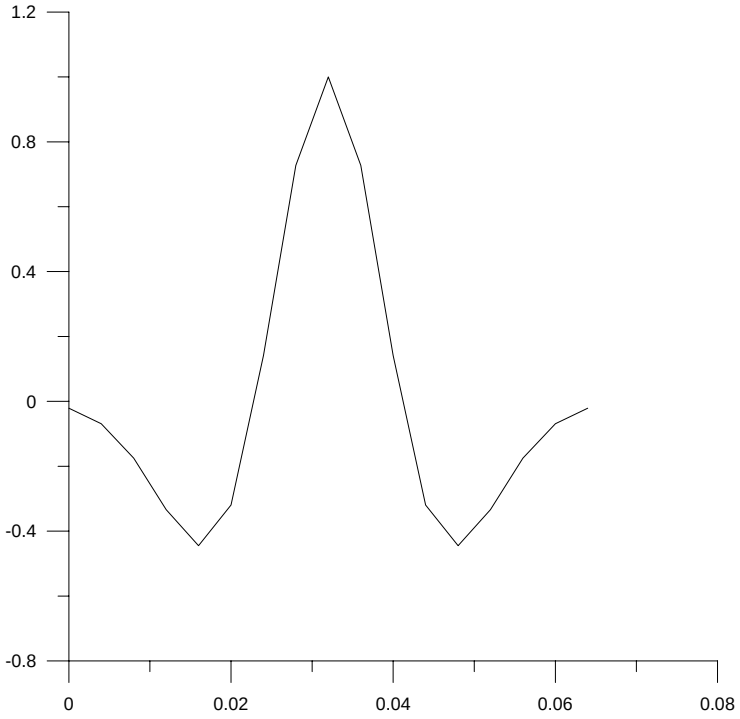


Fig.4.26. Forme de référence

$$\begin{pmatrix} 4.952445E\text{-}008 \\ 15.30085 \\ 53.97363 \\ 728.8595 \\ 4463.33 \end{pmatrix}$$

Fig.4.27. Matrice des moments invariants.

- Sur la figure (4.28), nous avons présenté une trace à trois pics. Nous avons calculé les coefficients d'inter-corrélation entre la matrice des moments du signal Ricker et une matrice des moments calculée le long de cette trace sur une fenêtre glissante dont la longueur est 20 échantillons.

Le résultat de cette application est présenté dans les figures (4.29) et (4.30) qui correspondent respectivement aux coefficients d'inter-corrélation et aux coefficients d'inter-corrélation normalisés. Nous remarquons que les pics des coefficients d'inter-corrélation correspondent bien aux positions des pics de la trace synthétique.

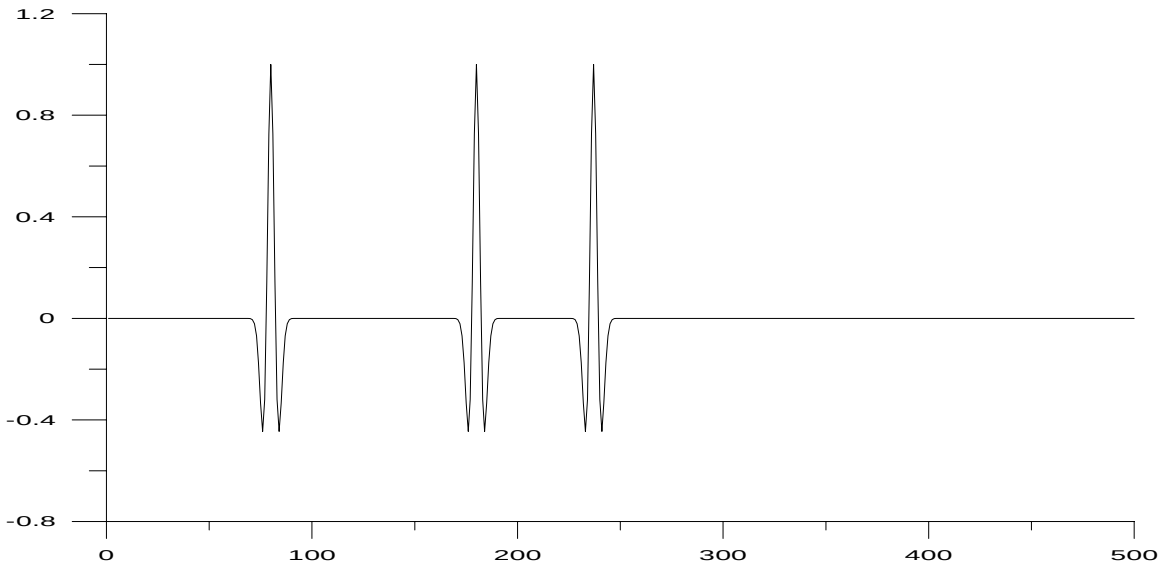


Fig.4.28. Trace synthétique.

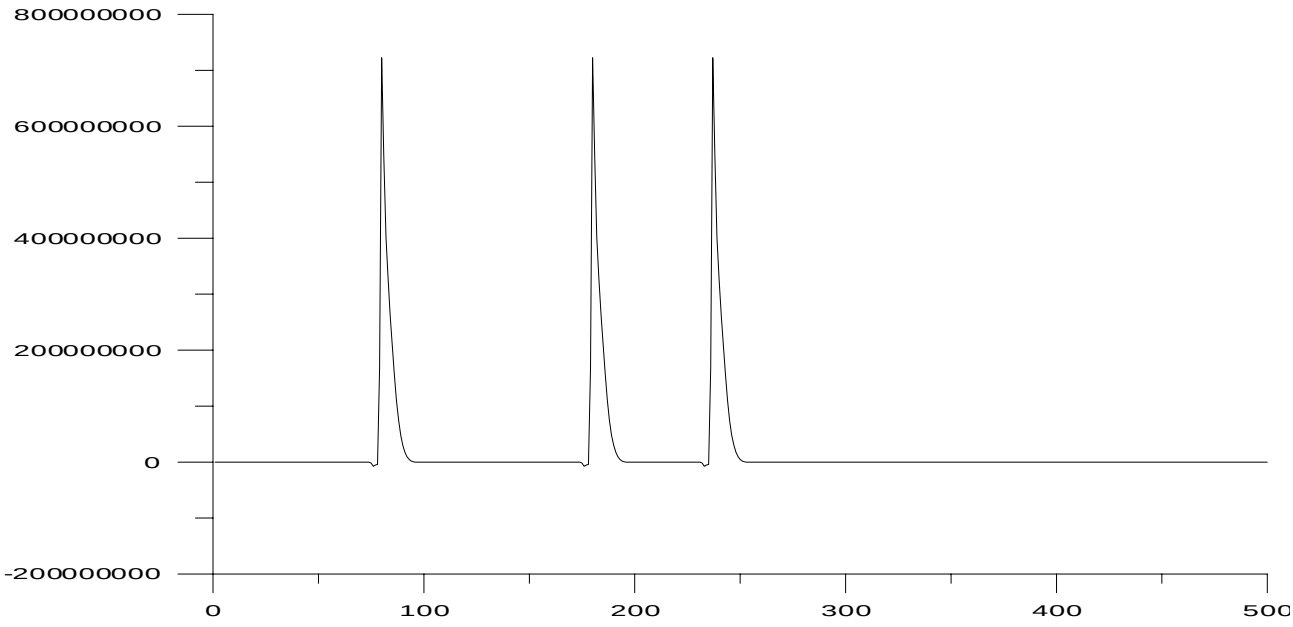


Fig.4.29. Coefficients d'inter-corrélation

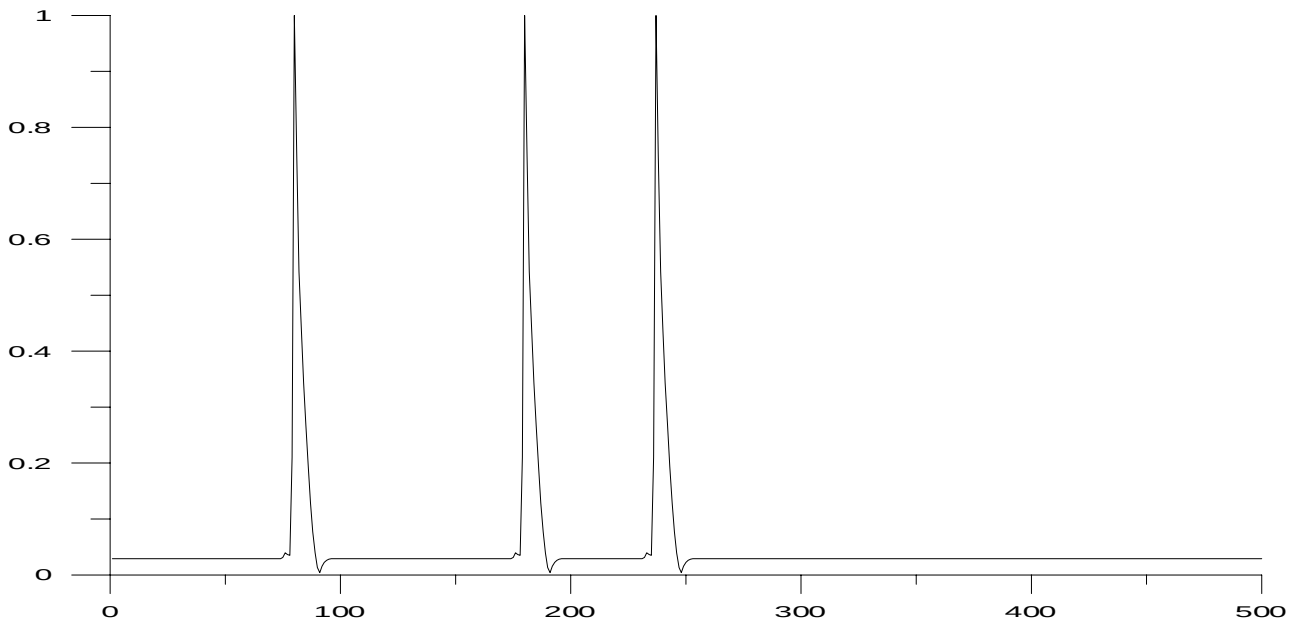


Fig.4.30. Coefficients d'inter-corrélation normalisés

- En vue de tester la performance de cette technique à la qualité des données, nous avons testé la même trace synthétique avec l'ajout du bruit de taux 5%, 30% et 50%. Les résultats obtenus sont schématisés respectivement dans les figures (4.32), (4.34) et (4.36). Nous constatons que les trois pics ont été bien repérés dans les trois situations ; ce qui montre la performance de cette technique.

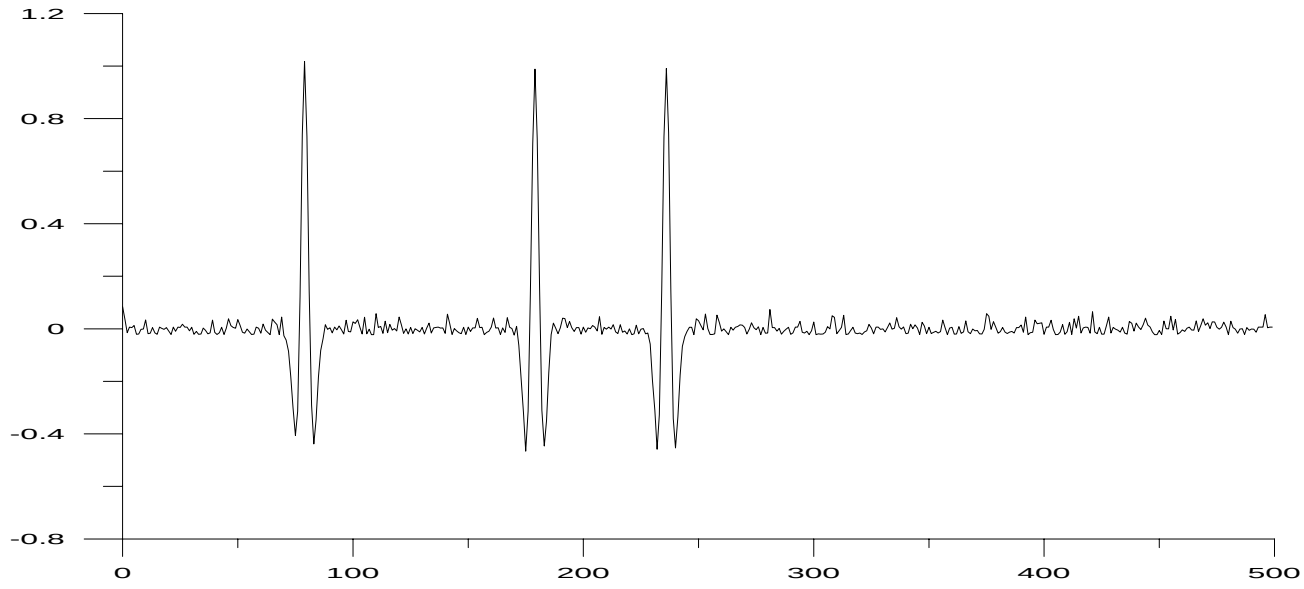


Fig.4.31. Trace bruitée (taux de bruit 5%)

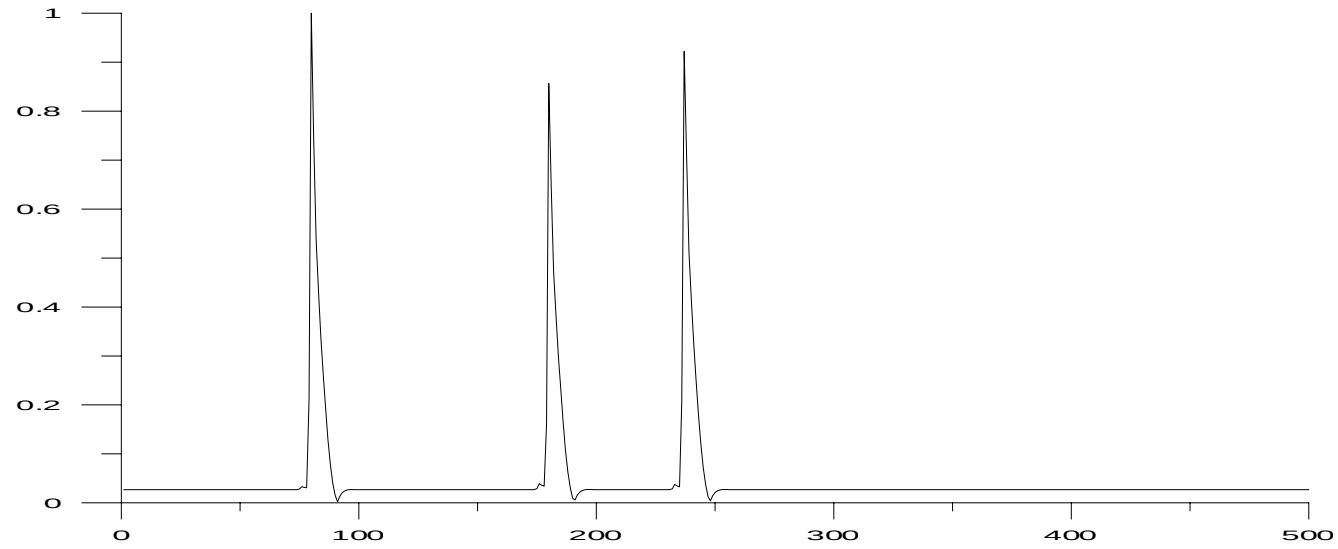


Fig.4.32. Coefficients d'inter-corrélation normalisés

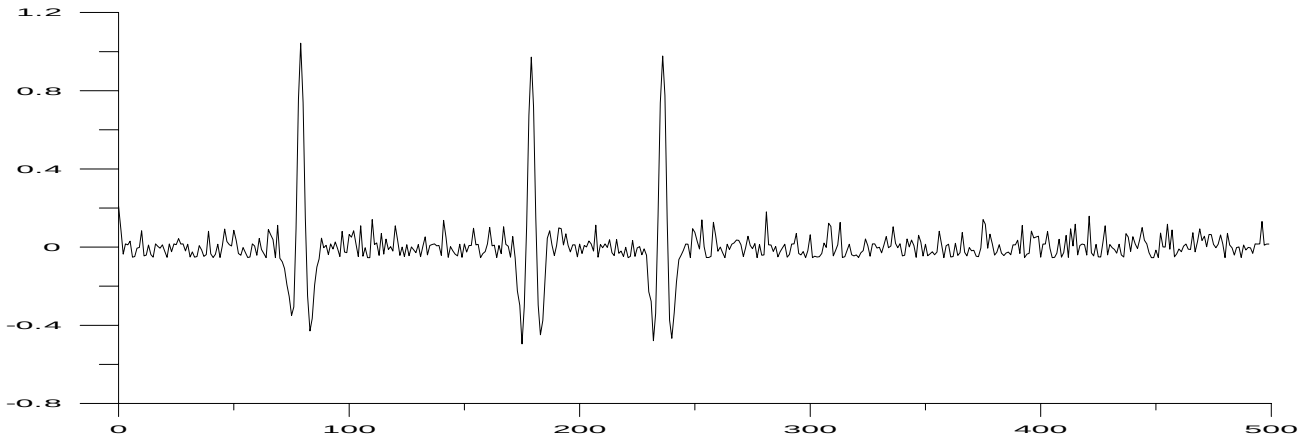


Fig.4.33. Trace bruitée (taux de bruit 30%)

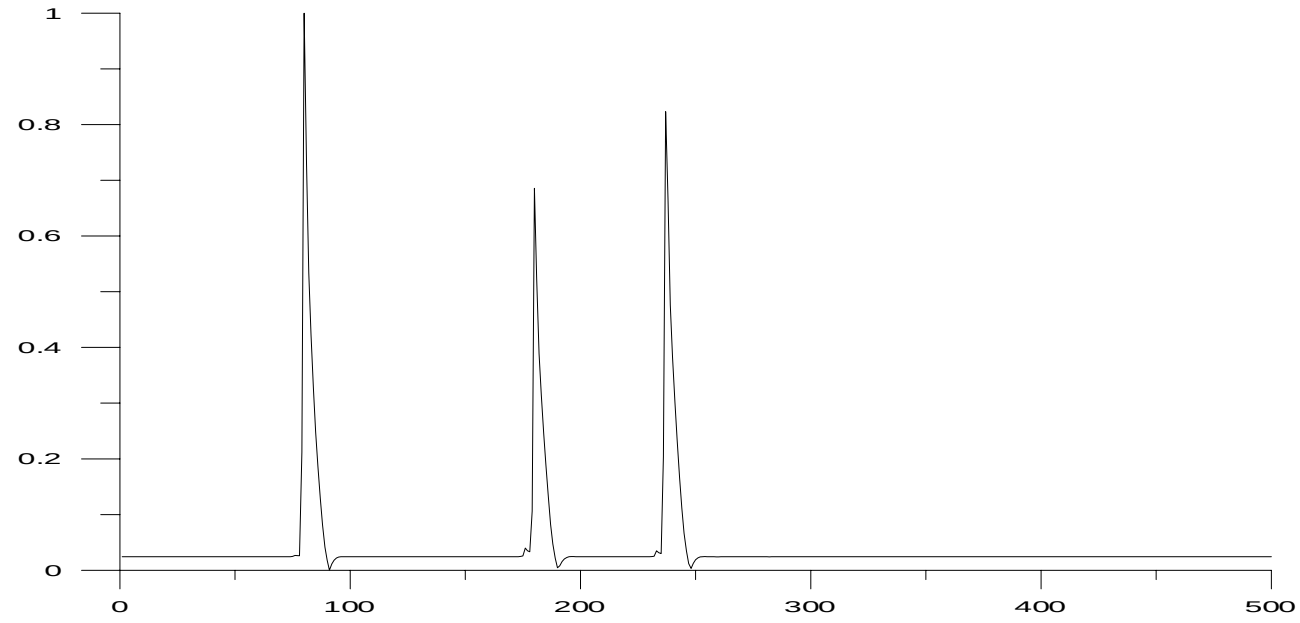


Fig.4.34. Coefficients d'inter-corrélation normalisés

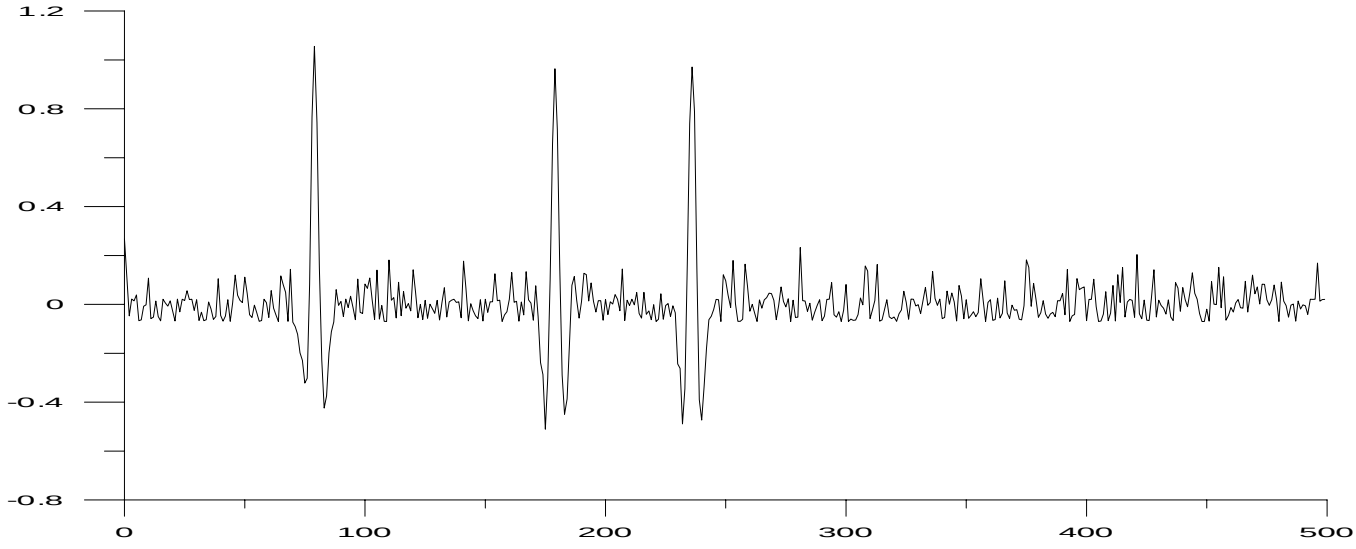


Fig.4.35. Trace bruitée (taux de bruit 50%)

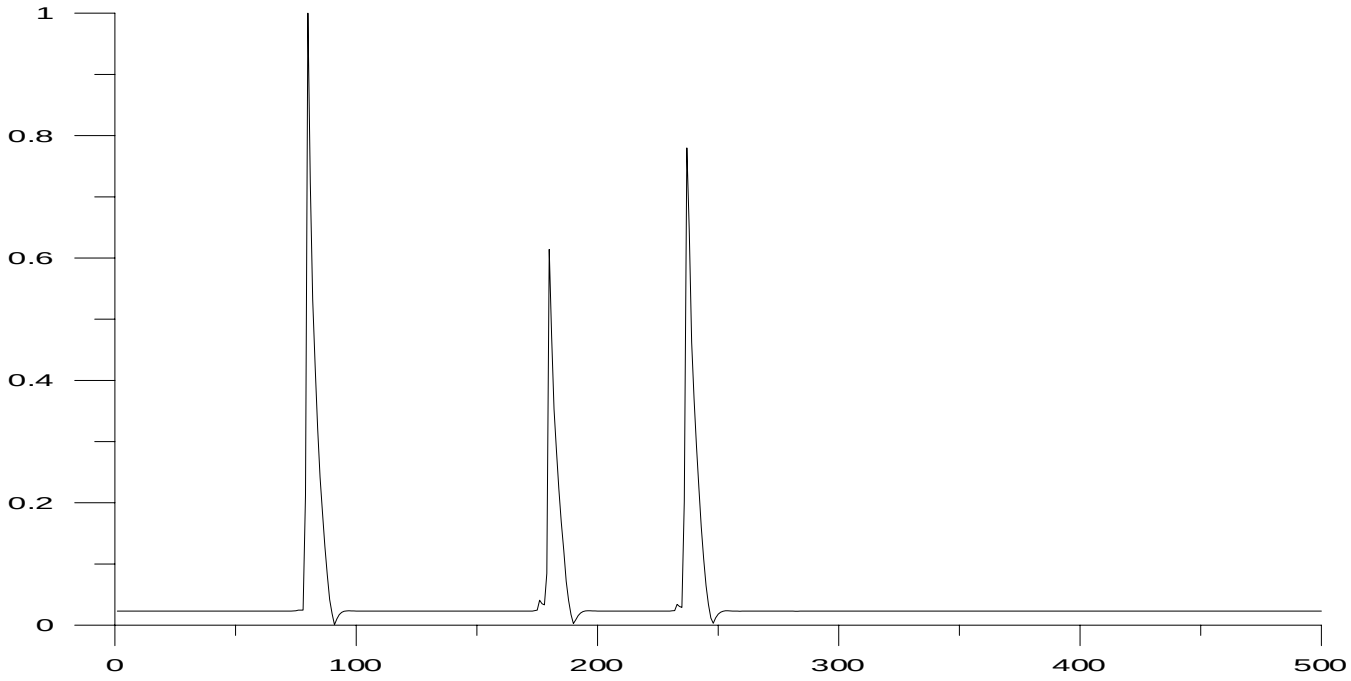


Fig.4.36. Coefficients d'inter-corrélation normalisés

Nous avons appliqué cette technique au film synthétique bruité présenté dans la figure (4.37) et dont les paramètres sont :

H1=100m, H2=300m, H3=400m :
V1=1000m/s, V2=1500m/s, V3=3500m/s, V4=4000m/s ;
Nombre de trace = 20 ;
Nombre d'échantillon =500 ;
Tau d'échantillonnage =0.004s.

Sur le résultat obtenu présenté dans la figure (4.38), nous constatons que les trois réflecteurs ont été bien pointés.

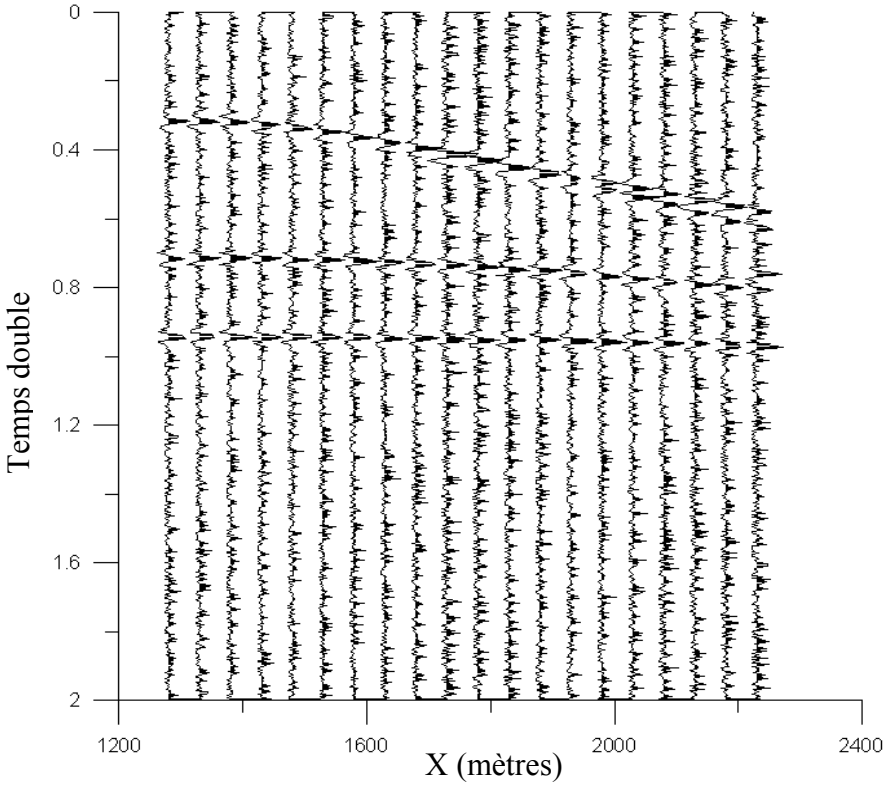


Fig.4.37. Film synthétique bruité (taux de bruit 5%).

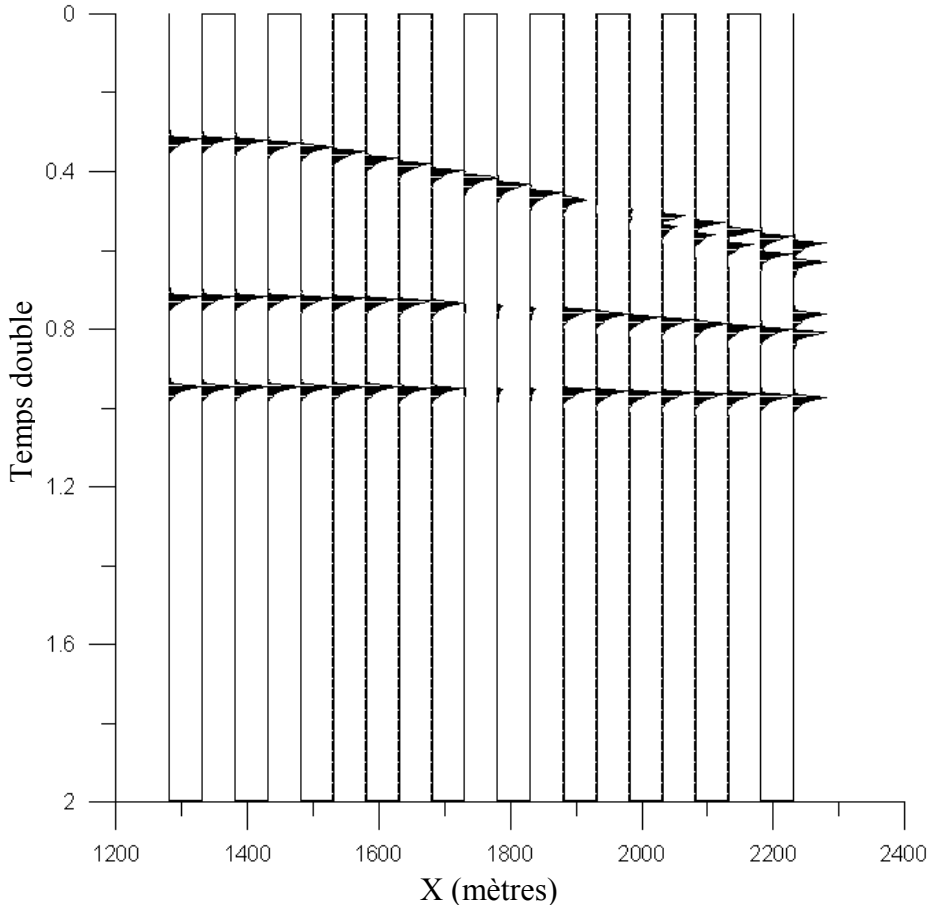


Fig.4.38. Film Output.

b. Cas réel :

Pour appliquer cette technique à la détection du même horizon indiqué dans le premier cas, nous avons pris une forme de référence comprise entre les échantillons 434 et 443 de la première trace de la section réelle. Cette forme de référence est présentée dans la figure (4.39) et sa matrice des moments invariants est présentée dans la figure (4.40).

Nous avons appliqué cette technique avec une fenêtre glissante de longueur 10 échantillons et sur un intervalle de la section (400 et 450 échantillons), c'est-à-dire sur la portion où il existe l'objectif ciblé à *priori*. Dans le résultat obtenu montré dans la figure (4.42), nous remarquons qu'à chaque trace il y 'a un pic et que l'horizon ciblé est bien repéré. Ce résultat est beaucoup plus performant que celui obtenu par la technique de l'intelligence artificielle.

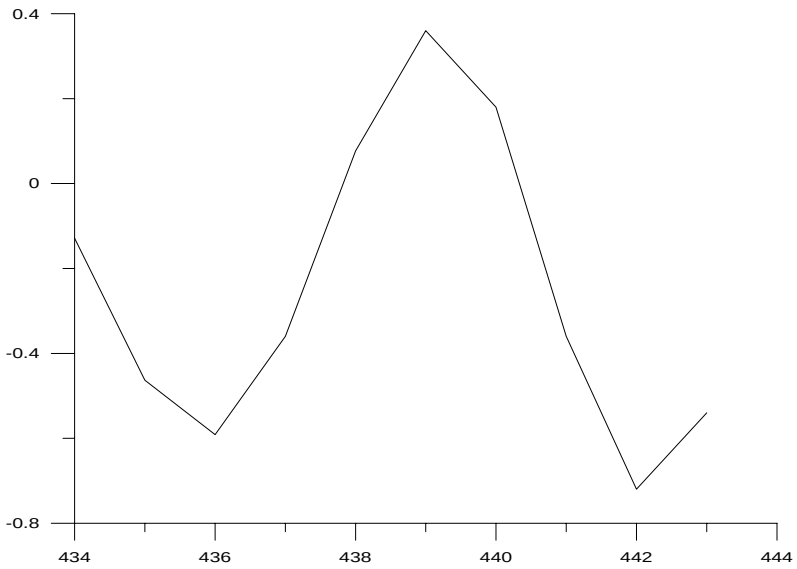


Fig.4.39 Forme de référence.

$$\begin{pmatrix} -0.1221594 \\ 5.340698 \\ -1.568622 \\ 65.42374 \\ -52.61512 \end{pmatrix}$$

Fig.4.40 Matrice référence des moments.

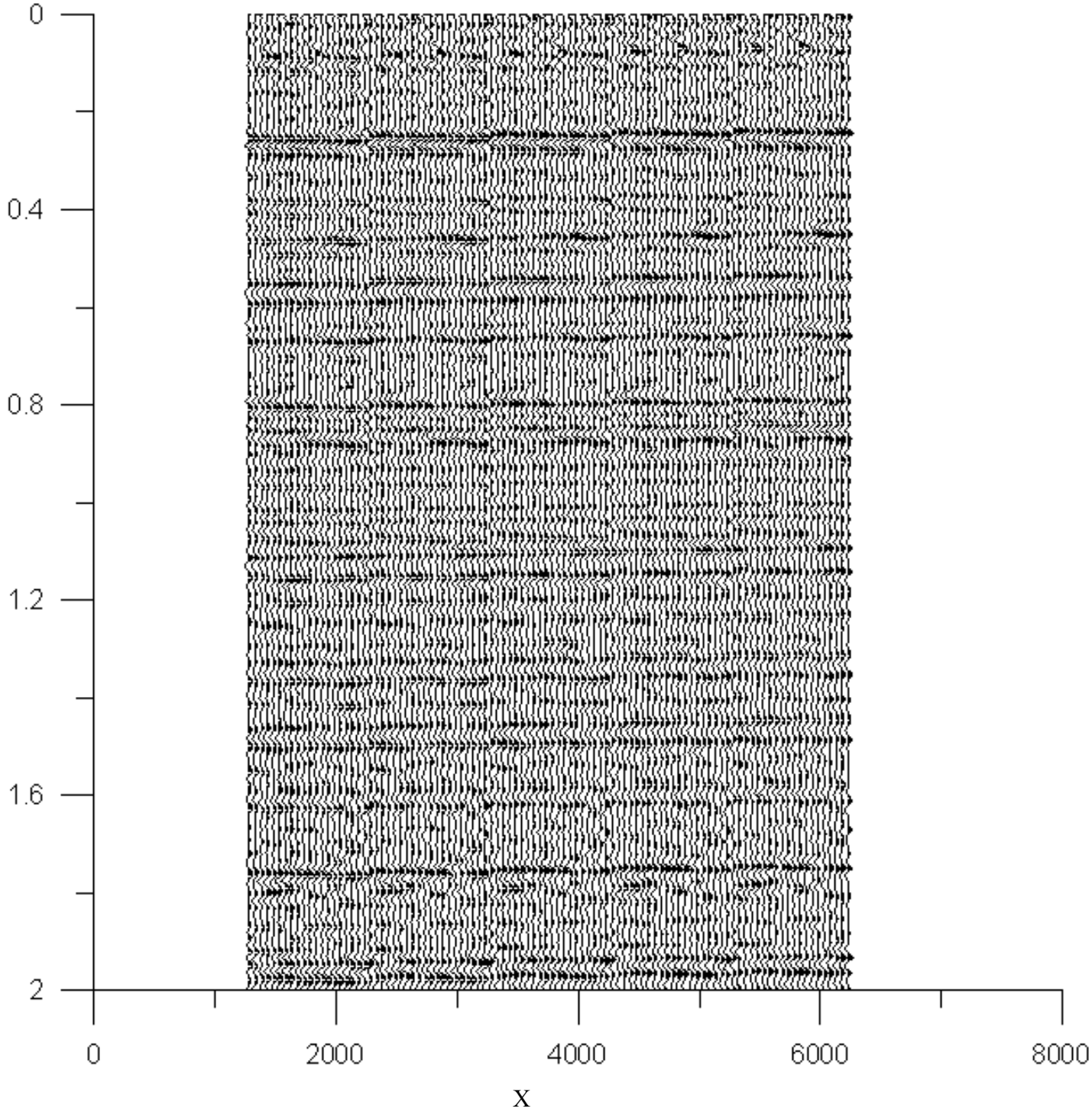


Fig.4.41 Section réelle.

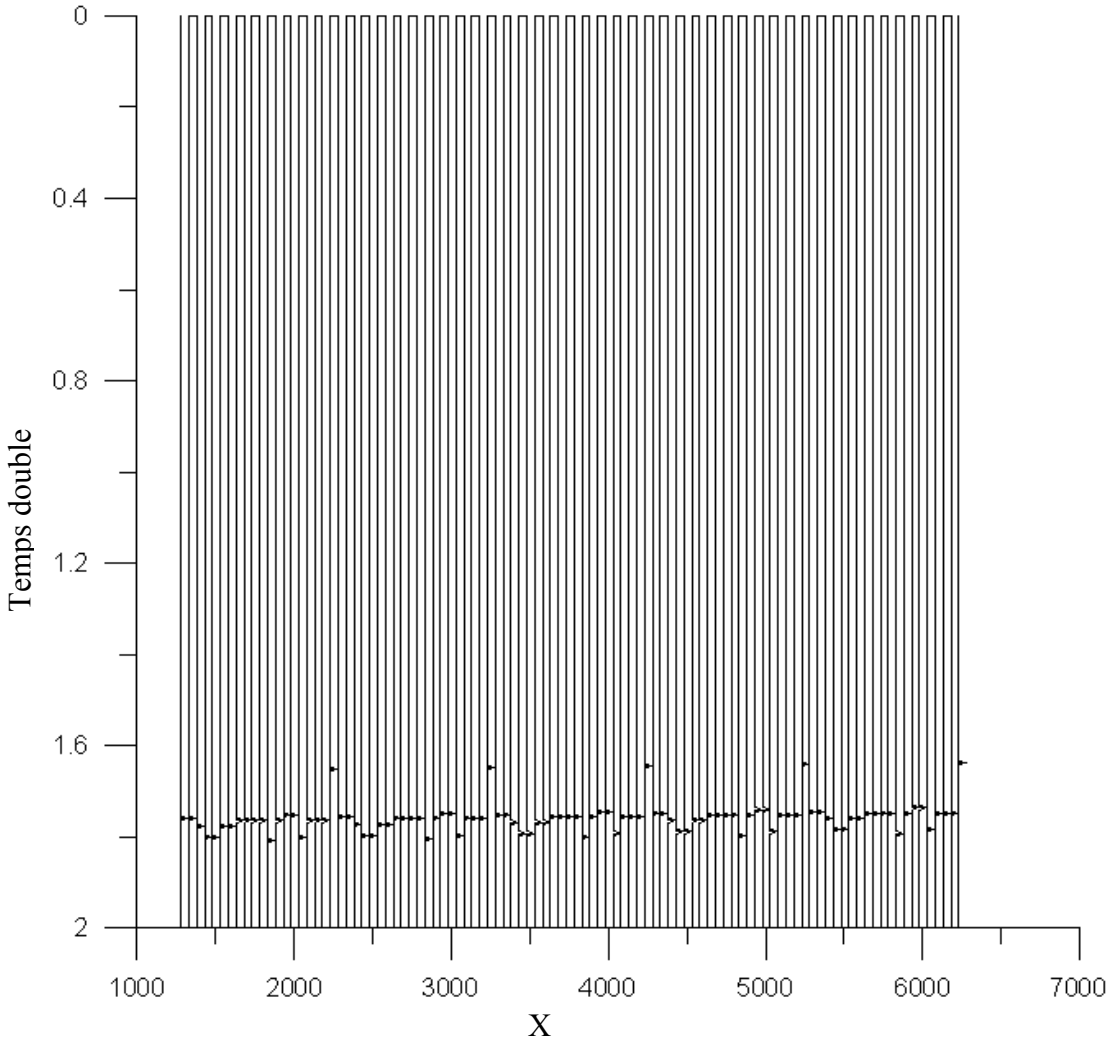


Fig.4.42 Section output

IV.2 APPLICATIONS ELECTRIQUES

IV.2.1 Inversion électrique par réseaux de neurones artificiels

a. Cas synthétiques

• Le tableau N°4.6 montre l'étendue des valeurs des modèles électriques utilisés dans l'étape d'apprentissage et la figure (4.43) montre la structure ou la topologie du réseau de neurones utilisé. Cette dernière est composée de deux couches, une couche cachée constituée de six neurones, et une couche output constituée de quatre neurones. Il est important de signaler qu'à l'input de réseau, on injecte les différentes valeurs de la résistivité apparente et à l'output, on aura les différents paramètres du modèle électrique, c'est à dire les épaisseurs et les résistivités.

N° de couche	$\rho_{min.}$	$\rho_{max.}$	$h_{min.}$(m)	$h_{max.}$(m)
1	1	1	1	10
2	0.03	0.2	3	20
3	0.15	0.61		

Tableau N°4.6. Paramètres du modèle de sous-sol.

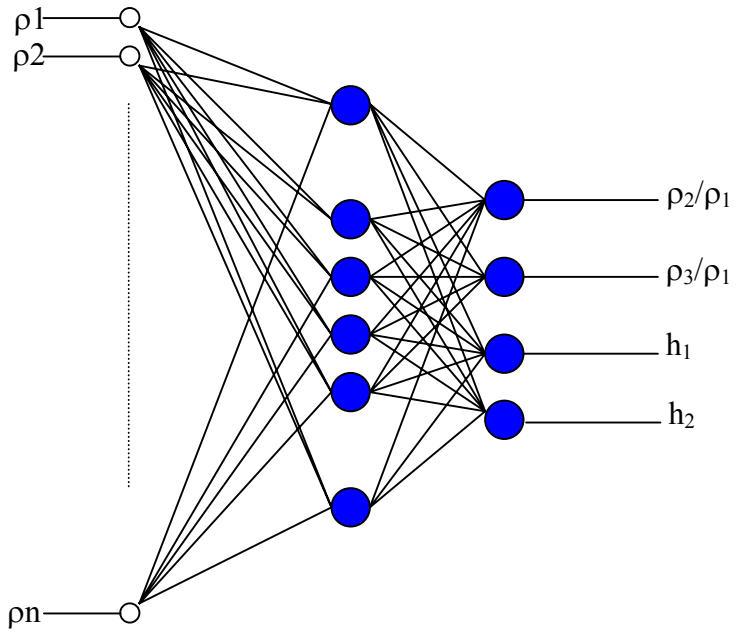


Fig.4.43 Réseau de neurones.

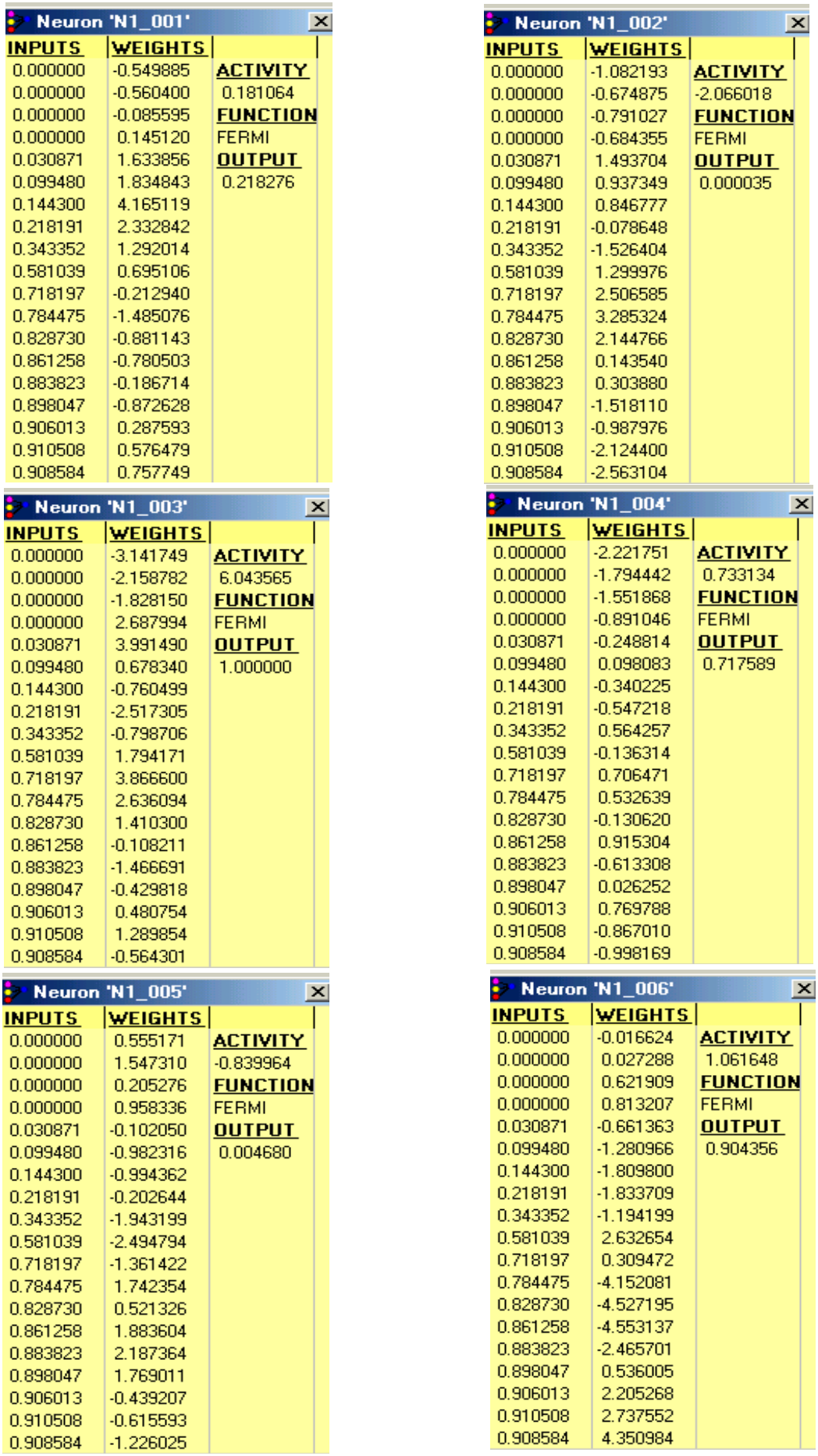
• L'étape d'apprentissage est réalisée par le logiciel Pythia, en utilisant cent exemples (tableau N°4.7) générés selon les caractéristiques montrées dans le tableau N°4.6. Après 1000 itérations, l'erreur quadratique moyenne passe de 0.4 à 0.042092. Nous avons obtenu les poids synaptiques de la couche cachée et la couchée output montrés respectivement dans les figures (4.44) et (4.45).

ρ2/ρ1	ρ3/ρ1	h1(m)	h2(m)
0,13223	0,5601412	1,000202	4,445551
0,1175459	0,3330839	9,711601	6,224726
0,04522312	0,4077794	3,366155	15,63971
0,1306262	0,3853878	6,240067	16,76263
0,153456	0,5946413	8,889705	19,91644
0,1829146	0,4503794	3,673921	10,24286
0,0580012	0,5446742	9,113809	19,34606
0,1891614	0,3407365	9,161601	7,998445
0,09722306	0,5133009	3,77611	11,75318
0,1307092	0,1930698	7,202272	12,25264
0,1319234	0,5386666	6,485362	10,34841
0,04218252	0,2382551	5,431438	16,13839
0,04427443	0,2334179	3,012472	16,26624
0,155683	0,2945005	6,550272	12,84541
0,1845112	0,6064605	4,949591	10,00374
0,1487487	0,3040302	6,193521	5,786681
0,05189554	0,230281	5,102042	4,29571
0,1157078	0,3650108	4,975996	3,634099
0,04587996	0,1806835	6,211143	14,45992
0,06244677	0,2260834	8,494324	9,626303
0,04930596	0,550494	9,626917	19,84367
0,1343877	0,4203447	6,600258	14,43509
0,1658197	0,4228009	2,35813	7,335245
0,06365159	0,2744213	9,069864	7,619242
0,1401742	0,4409759	4,567315	4,630814
0,04510475	0,5864767	2,125797	5,571638
0,1234954	0,488141	5,865725	3,724273
0,03123341	0,2523682	5,360904	10,91345
0,04955155	0,428102	2,85563	12,71523
0,07148543	0,4557554	2,531236	14,86603
0,09219382	0,4956875	1,372181	5,952087
0,1257916	0,4964241	4,540416	17,02359
0,1708345	0,3949736	7,233367	8,423838
0,1295073	0,5146089	8,023685	5,50842
0,1632178	0,5154153	9,5719	18,51303
0,05599608	0,3846243	6,148559	3,159943
0,1322464	0,4075627	7,851623	8,025092
0,1849455	0,2926075	8,624113	11,57093
0,04081879	0,5936494	8,141303	19,70384
0,1343445	0,2896164	2,999918	10,62558

0,07012298	0,5188828	9,257018	3,955543
0,1343471	0,1734888	4,927137	19,72481
0,1238841	0,1981618	8,574469	7,076658
0,0603913	0,3898472	9,541812	12,09155
0,1048179	0,2961135	8,516873	8,609651
0,06453367	0,4881398	7,365495	19,62359
0,03581464	0,1727554	4,258159	19,21149
0,1969046	0,1779889	8,916848	12,73315
0,0502537	0,6004258	1,617665	16,79769
0,1688125	0,33693	3,765615	6,839031
0,1971641	0,4970959	8,757817	13,52379
0,14331	0,1667864	2,690689	18,5718
0,1549737	0,5656757	5,653135	17,11979
0,08436424	0,4107201	7,684357	5,991867
0,1431035	0,3315019	2,569762	8,579091
0,09615734	0,2343034	3,26393	10,89738
0,1605828	0,4308526	5,91797	7,377495
0,1038805	0,2149946	8,371419	6,340564
0,1844471	0,5188434	4,01647	14,37885
0,07216797	0,3668569	3,720669	5,650229
0,07395888	0,6072298	3,706517	7,34738
0,1848981	0,5397815	3,688887	14,87894
0,04779625	0,2382216	5,308998	4,179483
0,07179807	0,3522404	7,358974	3,138493
0,03289538	0,2124022	5,426892	9,623424
0,138462	0,3622403	3,698685	7,236929
0,1901151	0,2451541	8,59152	14,87262
0,1927606	0,3314207	2,611312	6,09845
0,1658448	0,4791997	7,598771	5,42186
0,1923379	0,3163912	2,924598	11,04303
0,1938896	0,5971701	6,25651	3,00527
0,09899051	0,4509687	6,932054	7,124152
0,1011069	0,4242782	7,620548	18,98341
0,04979711	0,3025045	9,282906	18,36558
0,1106688	0,460683	4,22168	8,168736
0,1235259	0,3381549	2,151769	10,64634
0,06538246	0,4924725	4,611228	13,51119
0,09424323	0,4609768	1,208457	3,804861
0,1034909	0,3959504	9,635685	3,309365
0,1299073	0,3239005	3,596268	18,94896
0,08482599	0,4608454	6,258839	11,36928
0,1231719	0,5610936	2,514658	16,3312
0,1567607	0,2978966	9,165784	15,36102
0,1551474	0,2591041	8,389308	5,469029
0,1552213	0,2491315	1,455344	6,602692
0,05844457	0,4962954	6,039034	12,829
0,1024013	0,2113816	2,607324	16,43399
0,1898234	0,3218328	2,772253	9,467733
0,1302886	0,4466076	6,498016	8,176877
0,1108911	0,5100823	1,838607	16,98552
0,09671388	0,2462137	9,458096	11,91448

0,04805139	0,4442011	4,308253	13,77845
0,1442033	0,462963	6,059386	15,56209
0,1538752	0,1532594	4,485148	9,789903
0,05059491	0,5473945	1,260143	7,468481
0,06834386	0,4372141	4,240388	13,95142
0,08273374	0,4212006	4,349457	13,64259
0,06077199	0,4331132	9,690492	19,43059
0,08847787	0,4356677	1,35788	10,96076
0,05533973	0,2155481	1,83028	5,699499

Table N°4.6. Les données d'apprentissage.



Neuron 'N1_001'

INPUTS	WEIGHTS	
0.000000	-0.549885	ACTIVITY
0.000000	-0.560400	0.181064
0.000000	-0.085595	FUNCTION
0.000000	0.145120	FERMI
0.030871	1.633856	OUTPUT
0.099480	1.834843	0.218276
0.144300	4.165119	
0.218191	2.332842	
0.343352	1.292014	
0.581039	0.695106	
0.718197	-0.212940	
0.784475	-1.485076	
0.828730	-0.881143	
0.861258	-0.780503	
0.883823	-0.186714	
0.898047	-0.872628	
0.906013	0.287593	
0.910508	0.576479	
0.908584	0.757749	

Neuron 'N1_002'

INPUTS	WEIGHTS	
0.000000	-1.082193	ACTIVITY
0.000000	-0.674875	-2.066018
0.000000	-0.791027	FUNCTION
0.000000	-0.684355	FERMI
0.030871	1.493704	OUTPUT
0.099480	0.937349	0.000035
0.144300	0.846777	
0.218191	-0.078648	
0.343352	-1.526404	
0.581039	1.299976	
0.718197	2.506585	
0.784475	3.285324	
0.828730	2.144766	
0.861258	0.143540	
0.883823	0.303880	
0.898047	-1.518110	
0.906013	-0.987976	
0.910508	-2.124400	
0.908584	-2.563104	

Neuron 'N1_003'

INPUTS	WEIGHTS	
0.000000	-3.141749	ACTIVITY
0.000000	-2.158782	6.043565
0.000000	-1.828150	FUNCTION
0.000000	2.687994	FERMI
0.030871	3.991490	OUTPUT
0.099480	0.678340	1.000000
0.144300	-0.760499	
0.218191	-2.517305	
0.343352	-0.798706	
0.581039	1.794171	
0.718197	3.866600	
0.784475	2.636094	
0.828730	1.410300	
0.861258	-0.108211	
0.883823	-1.466691	
0.898047	-0.429818	
0.906013	0.480754	
0.910508	1.289854	
0.908584	-0.564301	

Neuron 'N1_004'

INPUTS	WEIGHTS	
0.000000	-2.221751	ACTIVITY
0.000000	-1.794442	0.733134
0.000000	-1.551868	FUNCTION
0.000000	-0.891046	FERMI
0.030871	-0.248814	OUTPUT
0.099480	0.098083	0.717589
0.144300	-0.340225	
0.218191	-0.547218	
0.343352	0.564257	
0.581039	-0.136314	
0.718197	0.706471	
0.784475	0.532639	
0.828730	-0.130620	
0.861258	0.915304	
0.883823	-0.613308	
0.898047	0.026252	
0.906013	0.769788	
0.910508	-0.867010	
0.908584	-0.998169	

Neuron 'N1_005'

INPUTS	WEIGHTS	
0.000000	0.555171	ACTIVITY
0.000000	1.547310	-0.839964
0.000000	0.205276	FUNCTION
0.000000	0.958336	FERMI
0.030871	-0.102050	OUTPUT
0.099480	-0.982316	0.004680
0.144300	-0.994362	
0.218191	-0.202644	
0.343352	-1.943199	
0.581039	-2.494794	
0.718197	-1.361422	
0.784475	1.742354	
0.828730	0.521326	
0.861258	1.883604	
0.883823	2.187364	
0.898047	1.769011	
0.906013	-0.439207	
0.910508	-0.615593	
0.908584	-1.226025	

Neuron 'N1_006'

INPUTS	WEIGHTS	
0.000000	-0.016624	ACTIVITY
0.000000	0.027288	1.061648
0.000000	0.621909	FUNCTION
0.000000	0.813207	FERMI
0.030871	-0.661363	OUTPUT
0.099480	-1.280966	0.904356
0.144300	-1.809800	
0.218191	-1.833709	
0.343352	-1.194199	
0.581039	2.632654	
0.718197	0.309472	
0.784475	-4.152081	
0.828730	-4.527195	
0.861258	-4.553137	
0.883823	-2.465701	
0.898047	0.536005	
0.906013	2.205268	
0.910508	2.737552	
0.908584	4.350984	

Fig.44. Poids synaptiques de la couche cachée.

Neuron 'N2_007'

INPUTS	WEIGHTS	
0.218276	0.227417	ACTIVITY
0.000035	0.407907	0.579179
1.000000	0.355108	FUNCTION
0.717589	0.457934	FERMI
0.004680	0.199521	OUTPUT
0.904356	-0.171531	0.578524

Neuron 'N2_008'

INPUTS	WEIGHTS	
0.218276	0.206998	ACTIVITY
0.000035	-0.553074	1.007609
1.000000	0.416155	FUNCTION
0.717589	0.263055	FERMI
0.004680	-0.308847	OUTPUT
0.904356	0.396935	0.883956

Neuron 'N2_009'

INPUTS	WEIGHTS	
0.218276	1.038961	ACTIVITY
0.000035	0.102169	-0.747309
1.000000	0.079731	FUNCTION
0.717589	-1.348289	FERMI
0.004680	-0.619672	OUTPUT
0.904356	-0.092226	0.006765

Neuron 'N2_010'

INPUTS	WEIGHTS	
0.218276	0.752181	ACTIVITY
0.000035	-0.544071	-0.127444
1.000000	-0.464471	FUNCTION
0.717589	-0.508492	FERMI
0.004680	-0.042154	OUTPUT
0.904356	0.594842	0.075176

Fig.4.45. Poids synaptiques de la couche output.

- Si à l'input du réseau obtenu, on injecte la courbe de la résistivité (figure.4.45) dont les paramètres sont montrés dans le tableau ci-dessous :

ρ2/ρ1	ρ3/ρ1	h1(m)	h2(m)
0,13223	0,5601412	1,000202	4,445551

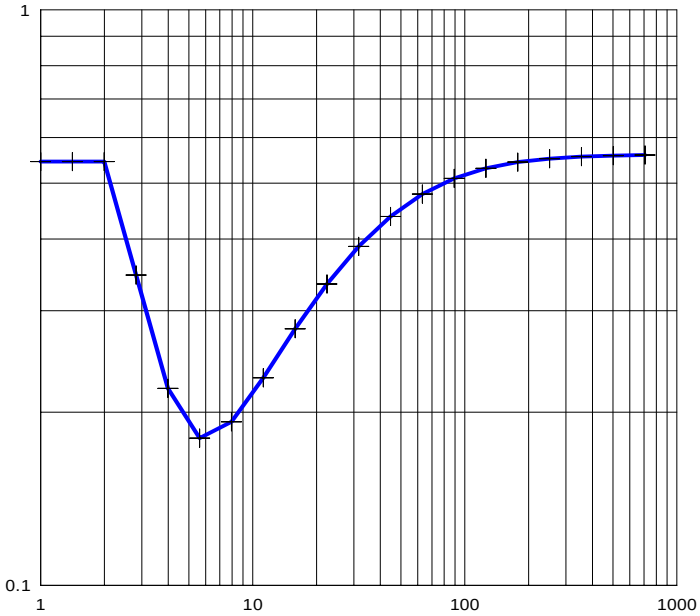


Fig.4.45. Courbe synthétique.

A l'output (Figure 4.46), nous aurons les paramètres suivants :

ρ2/ρ1	ρ3/ρ1	h1(m)	h2(m)
0.127228	0.554549	1.059133	4.276579

Dans la figure (4.47), nous avons présenté la courbe injectée et la courbe obtenue par le logiciel Pythia. L'erreur quadratique moyenne entre les deux courbes est de 0,00343391.

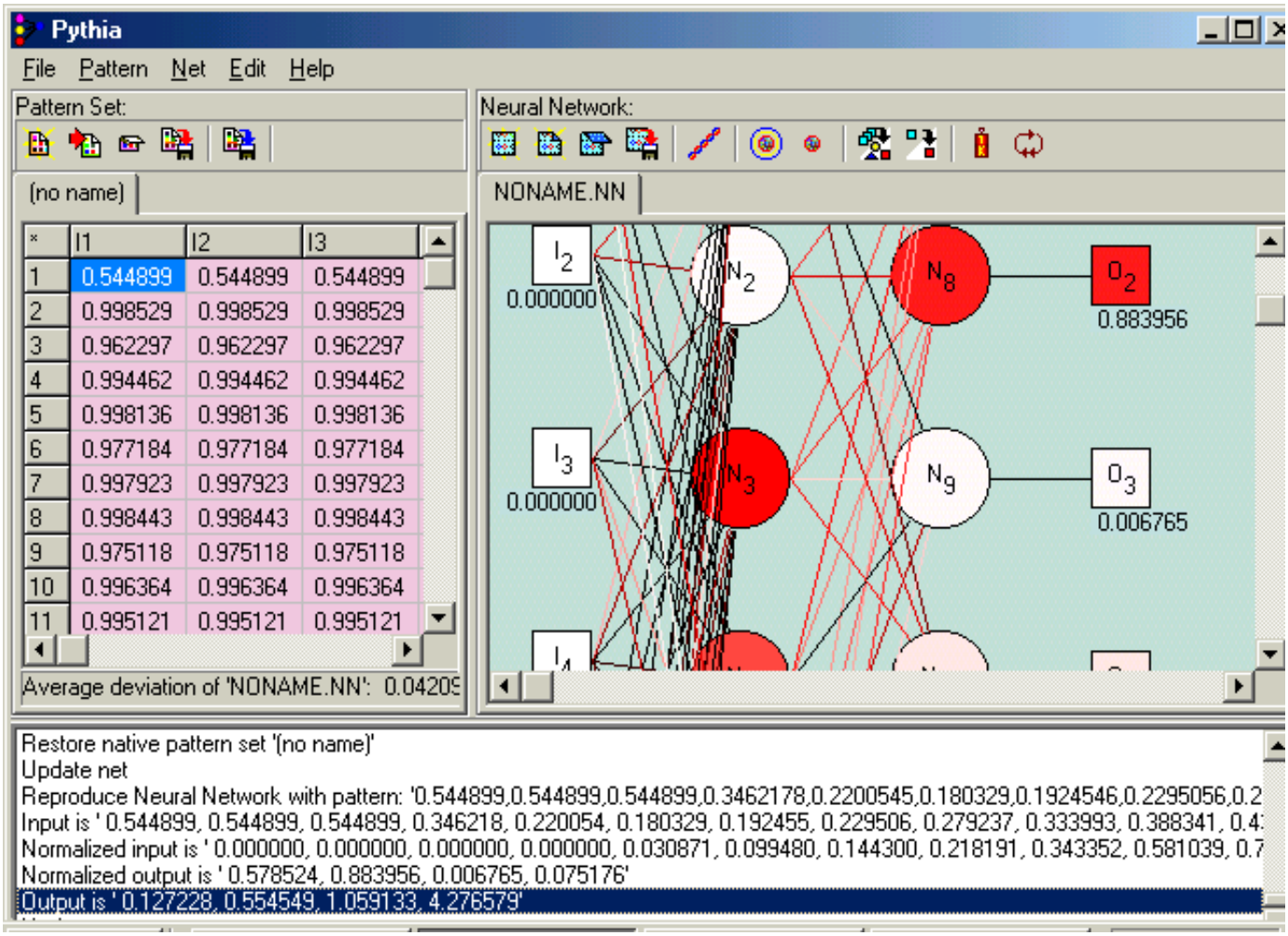


Fig.4.46 Ouput par le logiciel pythia

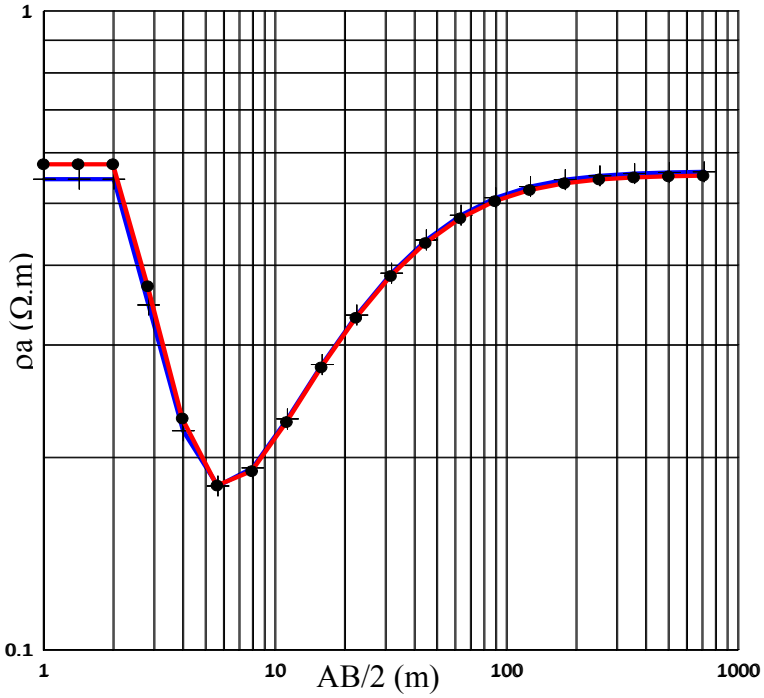


Fig.4.47 Courbe input et la courbe output.

b. Cas réel

• A l'input du réseau de neurones, nous avons injecté les données de la courbe réelle présentée dans la figure (4.48), à l'output nous aurons les résultats présentés dans la figure (4.49) et les paramètres suivants :

ρ2/ρ1	ρ3/ρ1	h1(m)	h2(m)
0.058593	0,400326	7.859529	19.345894

L'erreur quadratique moyenne entre les deux courbes est de 0,026234445.

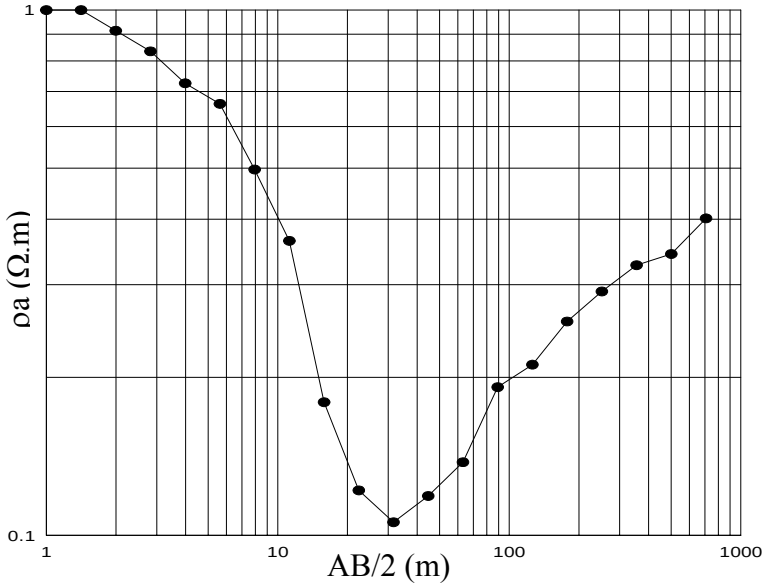


Fig.4.48 Courbe réelle

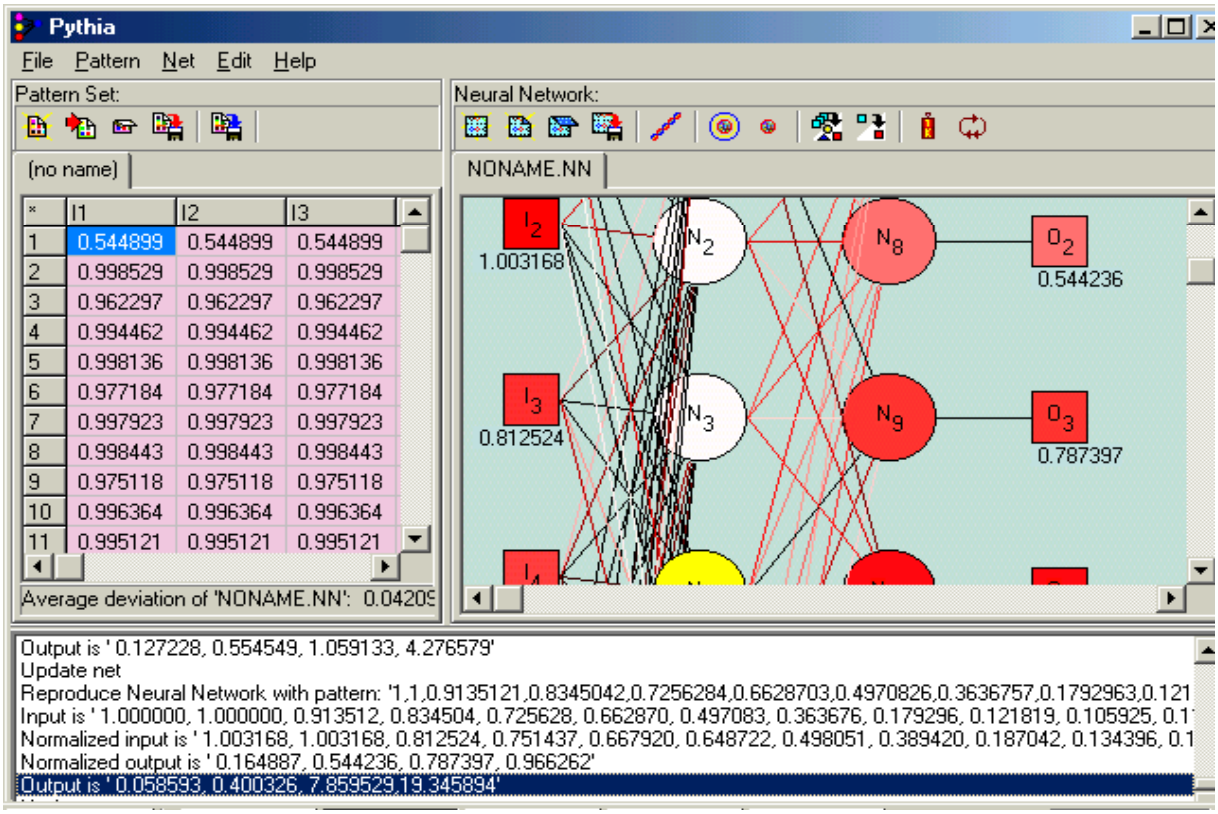


Fig.4.49 Output de la courbe réelle

IV.2.2 Inversion électrique par recuit simulé « Simulated Anuleaning »

a. Cas synthétiques

Après avoir appliqué la méthode recuit simulé pour l'inversion de l'exemple synthétique dont les paramètres sont les suivants :

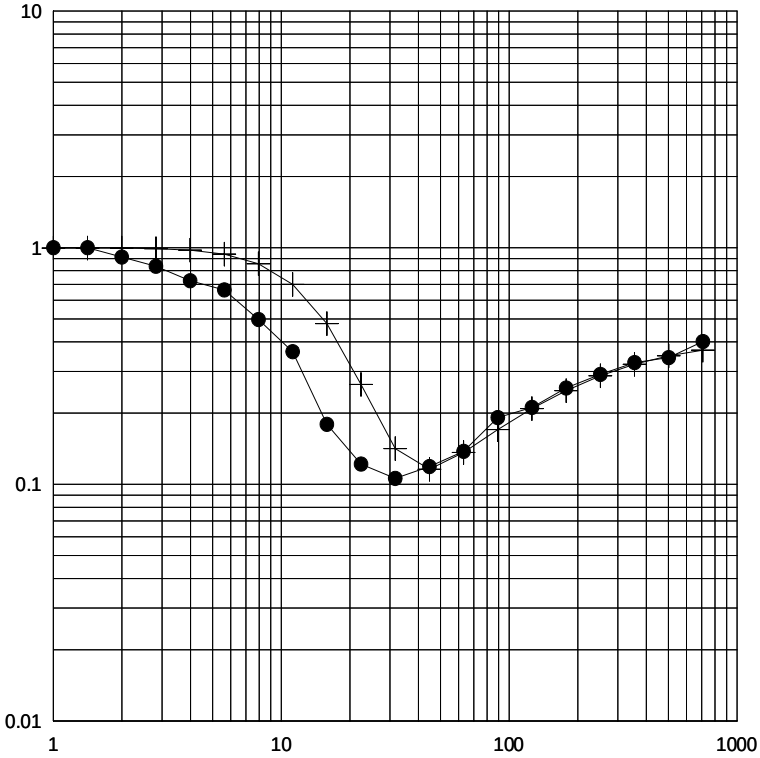
ρ2/ρ1	ρ3/ρ1	h1 (m)	h2 (m)
0,13223	0,5601412	1,000202	4,445551

Les paramètres utilisés sont :
Nombre de step =50 ;
Nombre d'essai =50 ;
Température initiale =100 ;
Paramètres initiaux :
h1=1 ;
h2=5 ;
r1=1 ;
r2=0.1 ;
r3=0.5 ;
dh=0.5 ;
dr=0.05.

Après 2500 itérations la courbe d'erreur obtenue est présentée dans le graphe (4.51), et le résultat obtenu est présenté dans le tableau ci-dessous :

ρ2/ρ1	ρ3/ρ1	h1 (m)	h2 (m)
0.1396047	0.557932	0.9796733	3.531791

Avec une erreur quadratique moyenne de 2.434569E-005



+ Courbe obtenue par pythia
- Courbe réelle

Fig.4.50 Courbe input et courbe ouput

Si on utilise les paramètres initiaux suivants :
h1=2.5 ;
h2=2.5 ;
r1=1 ;
r2=0.2 ;
r3=0.2 ;
dh=0.5 ;
dr=0.05.

Nous aurons le résultat montré dans le tableau ci-dessous :

ρ2/ρ1	ρ3/ρ1	h1(m)	h2 (m)
0.7432144	0.4154629	2.012748	0.0144450

Avec une erreur moyenne quadratique de 0.08244723

Nous constatons que si les données initiales sont proches du modèle initial les résultats sont bons et vice versa, donc la performance de la méthode recuit simulé est très étroitement liée au choix judicieux des paramètres initiaux.

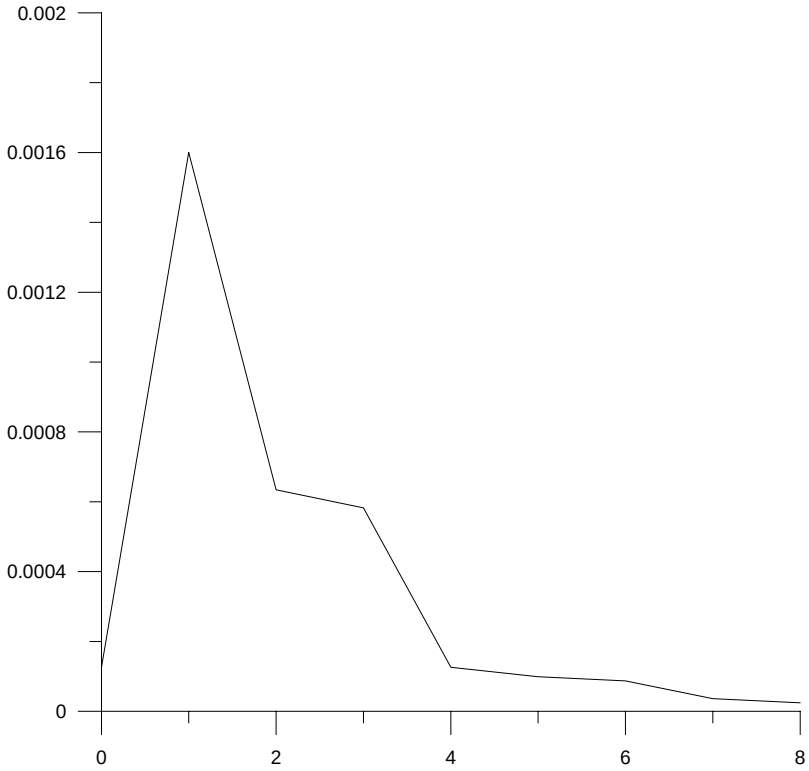


Fig.4.51 Courbe d'erreur de recuit simulé.

b. Cas réel

Après avoir injecté les données réelles présentées dans la figure (4.47), et ayant utilisé les paramètres suivants :

h1=5 ;
h2=10 ;
r1=1 ;
r2=0.1 ;
r3=0.2 ;
dh=0.5 ;
dr=0.05 ;

Nous aurons le résultat présenté dans le tableau ci-dessous :

ρ2/ρ1	ρ3/ρ1	h1 (m)	h2 (m)
0.044145	0.39016	4.018163	11.66403

Avec une erreur quadrique moyenne de 0.003200929

En vue d'améliorer le résultat, nous utilisons le résultat obtenu par réseau de neurones artificiels comme paramètre d'entrée dans la méthode recuit simulé. Nous obtenons le résultat montré dans le tableau ci-dessous :

ρ2/ρ1	ρ3/ρ1	h1(m)	h2 (m)
0.0621961	0.3807792	3.91353	16.08627

Avec une erreur quadratique moyenne de 0.003010727.

Conclusion & recommandations

CONCLUSION ET RECOMMANDATIONS

Dans l'application sismique, le réseau de neurones choisi est constitué de deux couches, une couche cachée contenant quatre neurones et une couche de sortie formée d'un seul neurone. Nous avons pris six entrants qui sont l'espérance mathématique de la transformé d'Hilbert, de l'amplitude instantanée, de la phase instantanée, de l'énergie sur une fenêtre, de l'entropie et du signal réel. Par ailleurs, nous avons choisi à la sortie du réseau une seule valeur égale à l'unité.
L'étape d'apprentissage est réalisée sur une forme de référence par le logiciel PYTHIA. Dans le cas synthétique, cette forme est le signal de Ricker ayant 17 échantillons et une fréquence centrale de 25Hz. Dans le cas réel, la forme de référence est choisie à un niveau de la section sismique où le réflecteur présente une bonne continuité.

L'application de la technique des réseaux de neurones artificiels pour la détection des horizons sismiques sur des traces synthétiques, sur des films synthétiques et sur une portion d'une section sismique réelle donne des résultats dont la performance est étroitement dépend de deux facteurs. Le premier est le choix judicieux de la longueur de la fenêtre glissante dans laquelle les attributs sismiques sont calculés, en effet la longueur adéquate est déterminée par test. Le second facteur est la qualité des données sismiques. En effet, cette technique exige un bon rapport signal sur bruit.

Par ailleurs, la technique des moments invariants donne des résultats plus performants que ceux obtenus par intelligence artificielle. Ceci est valable pour des applications synthétiques et réelles.

L'inversion des sondages électriques par réseaux de neurones artificiels donne des résultats probants sur un modèle synthétique à trois couches ainsi que sur des données réelles, ceci est obtenu après un processus d'apprentissage réalisé par le logiciel PYTHIA utilisant cent exemples d'apprentissage durant mille itérations.
En outre, le résultat obtenu par la technique recuit simulé « *Simulated Annealing* » est médiocre. Un bon résultat de cette technique est conditionné par un choix adéquat du modèle initial.
En vue de parfaire le résultat de la technique recuit simulé, nous avons utilisé le modèle obtenu par la technique d'intelligence artificielle en tant que modèle initial. En d'autres termes, nous avons appliqué les deux techniques en cascade.

A la lumière de ce mémoire, il nous parait primordial que l'étape d'apprentissage soit réalisée avec soin afin d'avoir un réseau puissant capable de détecter les horizons sismiques même en présence du bruit et d'inverser les courbes des sondages électriques avec une précision meilleure. Pour cela, nous recommandons les points suivants :

- Intégrer d'autres éléments caractérisant le signal sismique dans les entrants du réseau de neurones utilisé ;

- Augmenter le nombre de courbes de résistivité apparente constituant les données d'apprentissage ;
- Utiliser d'autres topologies des réseaux de neurones artificiels ;
- Utiliser d'autres algorithmes d'apprentissage autres que l'algorithme de rétro-propagation *«Back propagation »*.

Annexe

```
c--- Ce programme permet de réaliser un Réseau de neurone artificiel ----------

c Paramètres d'entrée:
c n: le nombre d'échantillon de signal input.
c m: le nombre des paramètres d'output.
c nom1:le fichier contenant les exemple d'apprentissage x.
c nom2:le fichier contenant les modèles d'apprentissage correspondant y.
c nneur: le nombre de neurone dans la couche cachée.
c niou:le taux d'apprentissage.
c nexemp: le nombre d'exemple d'apprentissage.
c alfa1:la constante de la fonction d'activation fh.
c alfa2:la constante de la fonction d'activation f0.
c nete: le nombre d'itération.
c les fichier o1.dat, h1.dat, b.dat, c.dat contiennent des données initiales.

c Paramètres de sorties.
c nom4: le fichier d'erreur.
c nom5: le fichier des poids Wh.
c nom6: le fichier des poids W0.

      real x(0:250,0:30),y(0:250,0:30)
      real wh(0:30,0:30),wo(0:30,0:30)
      real b(0:20),c(0:20)
      real ah(0:250,0:30),oo(0:250,0:30)
      real foprim(0:250,0:30),fhprim(0:250,0:30)
      real e(0:100000)
      real niou
      integer ete
      real alfa1,alfa2
      character*10 nom1,nom2,nom4,nom5,nom6
10    format(2x,"enter le nombre d'echantillon signal input N=....",$)
20    format(2x,"enter le nombre d'echantillon signal output M=....",$)
25    format(2x,"entrer le fichier de donnée input x(i), nom1=....",$)
30    format(2x,"entrer le fichier de donnée output desirés nom2=.",$)
35    format(20a)
40    format(2x,"entrer le nbre de neuron de la couche cachée nneu=.",$)
45    format(2x,"entrer le tau d'apprentissage niou=..",$)
50    format(2x,"entrer le nobre d'exmple d'apprentissage nexemp=...",$)
55    format(2x,"entrer la constante de la ftion d'activatin alfa1=",$)
56    format(2x,"entrer la constante de la ftion d'activatin alfa2=.",$)
60    format(2x,"enter le nombre d'iteration nete=...",$)
70    format(2x,"entrer le ficheir d'erreur nom4=...",$)
75    format(2x,"entrer le ficheir de Wh de FNN nom5=...",$)
80    format(2x,"entrer le ficheir de Wo de FNN nom6=...",$)

      write(6,10)
      read(5,*)n
      write(6,20)
      read(5,*)m
      write(6,25)
```

```
          read(5,35)nom1
          write(6,30)
          read(5,35)nom2
          write(6,40)
          read(5,*)nneur
          write(6,45)
          read(5,*)niou
          write(6,50)
          read(5,*)nexemp
          write(6,55)
          read(5,*)alfa1
          write(6,56)
          read(5,*)alfa2
          write(6,60)
          read(5,*)nete
          write(6,70)
          read(5,35)nom4
          write(6,75)
          read(5,35)nom5
         write(6,80)
          read(5,35)nom6

c la lecture des signaux d'entre x(i)
          open(10,file=nom1)
          do q=1,nexemp
          do i=1,n
          read(10,*)a,x(q,i)
          enddo
          enddo
          close(10)

c-----la lecteur des outputs désirés y(j)-----
          open(10,file=nom2)
          do q=1,nexemp
          do k=1,m
          read(10,*)a,y(q,k)
          enddo
          enddo
      close(10)

c---- initialisation des poids Wh et Wo et b() et c() ----

          open(10,file="h1.dat")
          do j=1,nneur
          do i=1,n
          read(10,*)wh(j,i)
          enddo
          enddo
          close(10)
```

```
          open(10,file="o1.dat")
          do k=1,m
          do j=1,nneur
          read(10,*)wo(k,j)
          enddo
          enddo
          close(10)

         open(10,file="b.dat")
         do j=1,nneur
          read(10,*)b(j)
          enddo
          close(10)

          open(10,file="c.dat")
          do k=1,m
          read(10,*)c(k)
          enddo
          close(10)

c----- Calcul -------------------

          do ete=1,nete
     call calculo(nexemp,x,wh,wo,alfa1,alfa2,b,c,n,m,nneur,ah,oo,
    *foprim,fhprim)

     call poids(foprim,fhprim,ete,nexemp,m,n,nneur,niou,
    *x,y,ah,oo,wh,wo,e)
          write(6,*)"ereur=",e(ete)
          write(6,*)"ete=",ete
          enddo
c-----------------------------------------
c---- fichier d'erreur ---
          open(10,file=nom4)
          do ete=1,nete
          write(10,*)ete,e(ete)
          enddo
          close(10)
c--- fichier des poids de FNN----
          open(10,file=nom5)
          do j=1,nneur
          do i=1,n
          write(10,*)wh(j,i)
          enddo
          enddo
          close(10)

          open(10,file=nom6)
          do k=1,m
          do j=1,nneur
```

```
          write(10,*)wo(k,j)
         enddo
          enddo
          close(10)
          end
          subroutine calculo(nexemp,x,wh,wo,alfa1,alfa2,b,c,n,m,nneur,ah
          *,oo,foprim,fhprim)
          real x(0:250,0:30)
          real wh(0:30,0:30),wo(0:30,0:30)
         real oo(0:250,0:30),ah(0:250,0:30)
          real b(0:20),c(0:20),alfa1,alfa2
          real foprim(0:250,0:30),fhprim(0:250,0:30)
          integer q

          do q=1,nexemp

c calcul de ahj
     do j=1,nneur
     sum1=0.
          do i=1,n
          sum1=sum1+wh(j,i)*x(q,i)
          enddo
          ah(q,j)=activ1(alfa1,(sum1+b(j)))
     fhprim(q,j)=derive1(alfa1,(sum1+b(j)))
     enddo

c calcul de ooj
          do k=1,1
          sum2=0.
          do j=1,nneur
          sum2=sum2+wo(k,j)*ah(q,j)
          enddo
          oo(q,k)=activ1(alfa2,(sum2+c(k)))
          foprim(q,k)=derive1(alfa2,(sum2+c(k)))
     enddo

     do k=2,2
          sum2=0.
          do j=1,nneur
          sum2=sum2+wo(k,j)*ah(q,j)
          enddo
          oo(q,k)=activ1(alfa2,(sum2+c(k)))
          foprim(q,k)=derive1(alfa2,(sum2+c(k)))
     enddo
          do k=3,3
           sum2=0.
          do j=1,nneur
          sum2=sum2+wo(k,j)*ah(q,j)
          enddo
          oo(q,k)=activ1(alfa2,(sum2+c(k)))
```

```
          foprim(q,k)=derive1(alfa2,(sum2+c(k)))
      enddo
          do k=4,4
           sum2=0.
          do j=1,nneur
          sum2=sum2+wo(k,j)*ah(q,j)
          enddo
          oo(q,k)=activ1(alfa2,(sum2+c(k)))
          foprim(q,k)=derive1(alfa2,(sum2+c(k)))
      enddo

          enddo

      end
c----- correction de wh et wo et calcul d'erreur ----------

          subroutine poids(foprim,fhprim,ete,nexemp,m,n,nneur,
     *niou,x,y,ah,oo,wh,wo,e)
          real x(0:250,0:30),y(0:250,0:30)
          real wh(0:30,0:30),wo(0:30,0:30)
      real ah(0:250,0:30),oo(0:250,0:30)
          real niou
          real e(0:100000)
      real foprim(0:250,0:30),fhprim(0:250,0:30)
      integer ete,q

          sum3=0.
      do q=1,nexemp
      do k=1,m
      sum3=sum3+(y(q,k)-oo(q,k))**2
          enddo
          enddo
          e(ete)=0.5*(1/float(nexemp))*sum3
          sum4=0.
          do k=1,m
      do j=1,nneur
      do q=1,nexemp
      sum4=sum4+(y(q,k)-oo(q,k))*foprim(q,k)*ah(q,j)
          enddo
      wo(k,j)=niou*sum4+wo(k,j)
      sum4=0.
      enddo
          enddo

          sum5=0.
      do j=1,nneur
          do i=1,n
      sum6=0.
          do q=1,nexemp
      sum5=0.
```

```
      do k=1,m
sum5=sum5+(y(q,k)-oo(q,k))*wo(k,j)*foprim(q,k)*fhprim(q,j)
      enddo
      sum6=sum6+sum5*x(q,i)*fhprim(q,j)
      enddo
      wh(j,i)=niou*sum6+wh(j,i)
      enddo
      enddo
      end

real function derive1(alfa,x)
      derive1=alfa/((cosh(alfa*x))**2 )
      end function
      real function activ1(alfa,x)
      activ1=tanh(alfa*x)
end function

real function derive2(alfa,x)
      derive2=(alfa*exp(-alfa*x))/((1+exp(-alfa*x))**2)
      end function
      real function activ2(alfa,x)
      activ2=0.17*(1/(1+exp(-alfa*x)))+0.03
end function
```

```
c ------------ Ce programme permet de calculer les données d'apprentissage ---------
c Paramètres d'entrée.
c nexemp: le nombre d'exemple d'apprentissage.
c n:nombre de couches
c hmax(),hmin rmax(),rmin() : les épaisseurs et les résistivités  max et min des modèles
utilisée.
c s1,s2,ds: tel que AB/2=s varie de 10**s1 à 10**s2 avec un pas de 10**ds.
c dl,lf: tel que la variable d'intégration  varie de 0 à lf avec un pas de dl
c le fichier de sortie: nom1: les inputs  x
c                       nom2: les otuputs  y.

      integer n, nexemp
          real r(0:100),h(0:100),s,t(0:1000000),kdl,rapp(0:1000),ds,s1,s2,dl
          real x(0:500,0:1000),y(0:500,0:30),a(0:10),b(0:10)
          real lf
          real hmin(0:10),hmax(0:10),rmin(0:10),rmax(0:10)
      character*20 nom1,nom2
10        format(2x,"entrer le nombre de couche n=...",$)
20        format(2x,"hmin et hmax(",i2,")=...",$)
21        format(2x,"hmin et  hmax(",i2,")=...",$)
25        format(2x,"rmin et  rmax(",i2,")=...",$)
26        format(2x,"entrer le pas de mesure ds=...",$)
30        format(2x,"entrer le pas d'echantillonnage dl=...",$)
35        format(2x,"entrer lamda finale lf=...",$)
40     format(2x,"entrer s1=....",$)
45     format(2x,"entrer s2=....",$)
50        format(2x,"entrer le nom de fichier de resultat x nom1=...",$)
51        format(2x,"entrer le nom de fichier de données y nom2=...",$)
55        format(10a)
60        format(2x,"entrer le nombre d'exemple utilisés nexemp=..",$)
          write(6,10)
          read(5,*)n
      write(6,60)
          read(5,*)nexemp
          do i=1,n-1
          write(6,21)i
          read(5,*)hmin(i),hmax(i)
      enddo
          do i=1,n
          write(6,25)i
          read(5,*)rmin(i),rmax(i)
          enddo
          write(6,26)
          read(5,*)ds
          write(6,30)
          read(5,*)dl
          write(6,35)
          read(5,*)lf
          write(6,40)
          read(5,*)s1
```

```
      write(6,45)
          read(5,*)s2
          write(6,50)
          read(5,55)nom1
          write(6,51)
          read(5,55)nom2

c le calcul.
          m =nint(lf/dl)
          do i=1,n-1
          a(i)=hmin(i)
          b(i)=hmax(i)
          enddo

          do j=1,nexemp


      do i=1,n-1
          call random (h(i))

      h(i)=(b(i)-a(i))*h(i)+a(i)
          a(i+1)=h(i)+hmin(i+1)
          b(i+1)=h(i)+hmax(i+1)
          y(j,i+n-1)=h(i)-h(i-1)
      enddo

          r(1)=1.
      do i=2,n
          call random(r(i))
          r(i)=(rmax(i)-rmin(i))*r(i)+rmin(i)
          y(j,i-1)=r(i)
      enddo

      do sp=s1,s2,ds
      s=10**sp
          sum=0.
          do k=1,m
      kdl=k*dl
          call fonctionr(n,r,h,kdl,t)
          sum=sum+(t(1)-r(1))*besj1(kdl*s)*k*(dl**2)
          enddo
          rapp(s)=r(1)+(s**2)*sum
          x(j,s)=rapp(s)
c      write(6,*)s
      enddo
          write(6,*)"exemple",j
      enddo

      open(10,file=nom1)
          do j=1,nexemp
```

```
do sp=s1,s2,ds
s=10**sp
      write(10,*)s,(x(j,s))
      enddo
      enddo
close(10)
      open(10,file=nom2)
      do j=1,nexemp
do i=1,2*n-2
write(10,*)i,y(j,i)
      enddo
      enddo
close(10)

end
      subroutine fonctionr(n,r,h,kdl,t)
      integer n
real t(0:1000000),r(0:10),e(0:10),h(0:10),kdl
      t(n)=r(n)
e(1)=h(1)
      do i=2,n-1
      e(i)=h(i)-h(i-1)
      enddo
do i=1,n-1
      t(i)=0.
      enddo
      do k=n-1,1,-1
t(k)=(t(k+1)+r(k)*tanh(kdl*e(k)))/(1+t(k+1)*tanh(kdl*e(k))/r(k))
      enddo
end
```

```
c------ Ce programme permet d'inverser les courbes de sondage électrique par
c         La méthode dite SIMULATED ANNEALING

c-------------------Les paramètres d'entrée----------------------
c s1,s2,ds: tel que AB/2=s varie de 10**s1 à 10**s2 avec un pas de 10**ds.
c dl,lf: tel que la variable d'intégration  varie de 0 à lf avec un pas de dl
c dl,lf,si,s2,ds sont utilises la modélisation
c (dans ces applications dl=0.001,lf=10,s1=0,s2=3,ds=0.15 pour une courbe de 20points )
c n:nombre de couche
c h(),r(),*:les paramètres initiaux
c dh(),dr( )( dh=0.05,dr=0.01)
c temp(0) : La temperature initiale.
c nstep : Nombre de step.
c nessai : nombre d'essai.
c nom1 le fichier de la courbe à inversée

c---------------Les Paramètres de sortie ---------------------------------
c nom2 le fichier d'erreur
c nom3 le fichier des épaisseurs
c nom4 le fichier des résistivités
c-------------------------------------------------------------------------

          integer n
          real r(0:100),h(0:100),s,t(0:1000000),kdl,ds,s1,s2,dl
          real r1(0:100,0:10),h1(0:100,0:10),rapp(0:100,0:100),temp(0:100)
      real de(0:100,0:100),reff(0:100)
          real r2(0:100,0:100),h2(0:100,0:100),erreur(0:100000)
      real finalh(0:1000,0:10),finalr(0:1000,0:10)
          real lf,dr(0:1000),dh(0:1000)
          character*20 nom1,nom2,nom3,nom4
10        format(2x,"entrer le nombre de couche n=...",$)
20        format(2x,"h(",i2,")=...",$)
25        format(2x,"r(",i2,")=...",$)
26         format(2x,"entrer le pas de mesure ds=...",$)
30        format(2x,"entrer le pas d'echantillonnage dl=...",$)
35        format(2x,"entrer lamda finale lf=...",$)
40     format(2x,"entrer s1=....",$)
45     format(2x,"entrer s2=....",$)
50        format(2x,"entrer le fichier des données à inversées nom1=...",$)
60        format(2x,"dh(",i2,")=...",$)
70        format(2x,"dr(",i2,")=...",$)
75        format(2x,"entrer le fichier erreur nom2=...",$)
80        format(2x,"entrer  le fichier des  epaisseur nom3=...",$)
85        format(2x,"entrer le fichier des resistivité nom4=...",$)
55        format(10a)
          write(6,26)
          read(5,*)ds
          write(6,30)
          read(5,*)dl
          write(6,35)
```

```
          read(5,*)lf
          write(6,40)
          read(5,*)s1
      write(6,45)
          read(5,*)s2
          write(6,50)
          read(5,55)nom1
          write(6,10)
          read(5,*)n

c-- Etape Initialisation ----------------
          write(6,*)'les paramètres initiaux'
          do i=1,n-1
          write(6,20)i
          read(5,*)h(i)
      h1(0,i)=h(i)
          enddo
          do i=1,n
          write(6,25)i
          read(5,*)r(i)
          r1(0,i)=r(i)
          enddo
          do i=1,n-1
          write(6,60)i
      read(5,*)dh(i)
          enddo
      do i=2,n
          write(6,70)i
          read(5,*)dr(i)
          enddo
          write(6,*)'entrer la température initiale'
          read(5,*)temp(0)
      write(6,*)'entrer le nombre de step'
          read(5,*)nstep
          write(6,*)'entrer le nombre d essais'
          read(5,*)nessai
          write(6,*)'---Les fichiers des résultats--------'
          write(6,75)
          read(5,55)nom2
          write(6,80)
          read(5,55)nom3
          write(6,85)
          read(5,55)nom4


c--- Lecture de Fichier à inverser ----------
          open(10,file=nom1)
      do sp=s1,s2,ds
          s=10**sp
          read(10,*)a,reff(s)
```

```
          enddo
          close(10)
c-------------------------------------------------------------

          do i=1,n-1
          h(i)=h1(0,i)
     enddo
          do i=1,n
          r(i)=r1(0,i)
          enddo

     m=nint(lf/dl)
     do sp=s1,s2,ds
     s=10**sp
          sum=0.
          do k=1,m
     kdl=k*dl
          call fonctionr(n,r,h,kdl,t)
          sum=sum+(t(1)-r(1))*besj1(kdl*s)*k*(dl**2)
          enddo
          rapp(0,s)=r(1)+(s**2)*sum
     enddo
          sum=0.
c ----------Calcul de DE---------------------
          sum1=0.
          do sp=s1,s2,ds
     s=10**sp
          sum1=sum1+(rapp(0,s)-reff(s))**2
     enddo
          de(1,0)=sum1
c------------Boucle de step-------------------
          do st=1,nstep
          write(6,*)'step=',st
          Temp(st)=0.9*Temp(0)

c----------Boucle d'essais-------------------

     do l=1,nessai
c- Perturbation aléatoire des paramètres
          write(6,*)'essais=',l
     do i=1,n-1
c         h2(l,i)=h1(l-1,i)+rh(temp,l,st)*dh(i)
          call random(f1)
          call random(f2)
     h2(l,i)=h1(l-1,i)+(-f1+f2)*dh(i)
     enddo
          r2(l,1)=1.
     do i=2,n
          call random(f1)
          call random(f2)
```

```
c       r2(l,i)=r1(l-1,i)+rr(temp,l,st)*dr(i)
        r2(l,i)=r1(l-1,i)+(-f1+f2)*dr(i)
    enddo


c---Le calcule.

    do i=1,n-1
        h(i)=h2(l,i)
    enddo
        do i=1,n
        r(i)=r2(l,i)
        enddo

        m =nint(lf/dl)
    do sp=s1,s2,ds
    s=10**sp
        sum=0.
        do k=1,m
    kdl=k*dl
        call fonctionr(n,r,h,kdl,t)
        sum=sum+(t(1)-r(1))*besj1(kdl*s)*k*(dl**2)
        enddo
        rapp(l,s)=r(1)+(s**2)*sum
    enddo

c -----------Calcul de DE----------------------
        sum1=0.
        do sp=s1,s2,ds
    s=10**sp
        sum1=sum1+(rapp(l,s)-reff(s))**2
    enddo
        de(st,l)=sum1
        dde=de(st,l)-de(st,0)
        write(6,*)'dde',dde

c-----Algorithme de Metrplis ------------
                if(dde.gt.0.)then
        p=exp(-dde/temp(st))
        call random(c)
                if(p.lt.c)then
                nombs=nombs+1
    erreur(nombs)=de(st,l)
                de(st,0)=erreur(nombs)
                do i=1,n-1
                finalh(nombs,i)=h2(l,i)
        enddo
                do i=1,n
                finalr(nombs,i)=r2(l,i)
```

```
        enddo
                write(6,*)'ok1 nombs=',nombs
        do i=1,n-1
        h1(l,i)=h2(l,i)
    enddo
        do i=1,n
        r1(l,i)=r2(l,i)
        enddo
                endif
                                if(p.gt.c)then

        do i=1,n-1
                h1(l,i)=h1(l-1,i)
       enddo
           do i=1,n
           r1(l,i)=r1(l-1,i)
           enddo
                                endif
                endif
                        if(dde.lt.0.)then
        write(6,*)'dde=',dde
    nombs=nombs+1
         do i=1,n-1
                finalh(nombs,i)=h2(l,i)
        enddo
                do i=1,n
                finalr(nombs,i)=r2(l,i)
        enddo


        erreur(nombs)=de(st,l)
        de(st,0)=erreur(nombs)

        write(6,*)'ok2 nombs=',nombs

    do i=1,n-1
        h1(l,i)=h2(l,i)
    enddo
        do i=1,n
        r1(l,i)=r2(l,i)
        enddo
                        endif



    enddo

c-----initialisation----------
        de(st+1,0)=erreur(nombs)
        do i=1,n-1
```

```
      h1(0,i)=finalh(nombs,i)
      enddo
do i=1,n
      r1(0,i)=finalr(nombs,i)
      enddo

enddo
      open(10,file=nom2)
      write(10,*)0,0.05*de(1,0)
do i=1,nombs
      write(10,*)i,0.05*erreur(i)
      enddo
      close(10)


      open(10,file=nom3)
      do j=1,nombs
      write(10,*)j,finalh(j,1)
      do i=2,n-1
      write(10,*)j,finalh(j,i)-finalh(j,i-1)
      enddo
      enddo
      close(10)
      open(10,file=nom4)
       do j=1,nombs
       do i=1,n
       write(10,*)j,finalr(j,i)
       enddo
       enddo
close(10)

      end

      subroutine fonctionr(n,r,h,kdl,t)
      integer n
real t(0:1000000),r(0:10),e(0:10),h(0:10),kdl
      t(n)=r(n)
e(1)=h(1)
      do i=2,n-1
      e(i)=h(i)-h(i-1)
      enddo
do i=1,n-1
      t(i)=0.
      enddo
      do k=n-1,1,-1
t(k)=(t(k+1)+r(k)*tanh(kdl*e(k)))/(1+t(k+1)*tanh(kdl*e(k))/r(k))
      enddo
end
      function rh(temp,l,st)
      real temp(0:100)
```

```
call random(b)
     a=(1+(1/(temp(st)**l)))**(2*b-1)
     c=b-0.5
if(c.gt.0)then
     rh=(temp(st)**l)*a
     else
     rh=-1.*(temp(st)**l)*a
     endif
write(6,*)'rh=',rh

end function

     function rr(temp,l,st)
     real temp(0:100)
     call random(b)
 a=(1+(1/(temp(st)**l)))**(2*b-1)
     c=b-0.5
     if(c.gt.0)then
     rr=(Temp(st)**l)*a
     else
     rr=-1.*(Temp(st)**l)*a
     endif
     write(6,*)'rr=',rr
end function
     function sgn(c)
     if(c.gt.0)then
     sgn=1.
     else
     sgn=-1.
     endif
 endfunction
```

```
c-- Programme de calcul des entrants de réseau de neurone ( attributs sismiques ) --

      complex tfs(0:3750),s(0:3750)
      character*10,nom1,nom4
5      format(10a)
10     format(1x,'entrer le signal … transformer nom1=...',$)
20     format(1x,'entrer la longueure de la fenetre  nech=...',$)
30    format(1x,'le pas d''echantillonnage tau=...',$)
80     format(1x,'entrer le fichier des Attributs nom4=...',$)
      write(6,10)
      read(5,5)nom1
      write(6,20)
      read(5,*)nech
      write(6,30)
      read(5,*)tau
      write(6,80)
      read(5,5)nom4

      call tf(nom1,nech,tau,nechf,tfs,sum6,sum8,sum9)
          call tfinv (tfs,nech,tau,s)

          open(10,file=nom4)
          sum1=0.
          sum2=0.
          sum3=0.
          sum4=0.
          sum5=0.
          do i=0,nech
          sum1=sum1+imag(s(i))
          sum2=sum2+sqrt(real(s(i))**2+imag(s(i))**2)
          sum3=sum3+atan(imag(s(i))/real(s(i)))
          sum4=sum4+sqrt(real(s(i))**2+imag(s(i))**2)*
    * atan(imag(s(i))/real(s(i)))
      sum5=sum5+cos(atan(imag(s(i))/real(s(i))))
          enddo
          write(10,*)sum1/nech
          write(10,*)sum2/nech
          write(10,*)sum3/nech
          write(10,*)sum6
          write(10,*)sum8/nech
          write(10,*)sum9/nech
          close(10)
          end


          subroutine tf(nom1,nech,tau,nechf,tfs,sum6,sum8,sum9)
      dimension as(0:3750),ps(0:3750),s(0:3750),entropie(0:1000)
      complex tfs(0:3750)
      character*10,nom1
c-------la lecture de signal ----------
```

```
      open(10,file=nom1)
      do k=0,nech
      read(10,*)a,s(k)
      enddo

c-- Caclcul de l'énergie et l'entropie
           sum6=0.
           do i=0,nech
           sum6=sum6+s(i)**2
           enddo

           sum7=0.
           do i=0,nech
           sum7=sum7+s(i)
           entropie(i)=sum7/sum6*log(s(i)**2/sum6)
           enddo
           sum8=0.
           do i=0,nech
           sum8=sum8+entropie(i)
           enddo
          sum9=0.
     do i=0,nech
          sum9=sum9+s(i)
          enddo
          close(10)
c-------------------------------------
      pi=atan(1.)*4
      fn=1/(tau)
      df=fn/nech
           nechf=nint(fn/df)
           do j=0,nechf
      sum1=0.
      sum2=0.
      do i=0,nech
      sum1=sum1+s(i)*cos(2*pi*j*i*tau*df)
      sum2=sum2+s(i)*sin(2*pi*j*i*tau*df)
      enddo
      as(j)=sqrt(sum1**2+sum2**2)
      ps(j)=atan(-sum2/sum1)
           tfs(j)=2*tau*cmplx(as(j)*cos(ps(j)),as(j)*sin(ps(j)))*
         *  cmplx(as(j)*cos(ps(j)-pi/2.),as(j)*sin(ps(j)-pi/2.))
           enddo
           end

       subroutine tfinv (tfs,nech,tau,s)
      complex tfs(0:3000),s(0:3000)
      complex sum1
      pi=atan(1.)*4
      fn=1./(tau)
      df=fn/nech
```

```
      nechf=nint(fn/df)
      do i=0,nech
      sum1=cmplx(0,0)
      do j=0,nechf
      sum1=sum1+(1./nech)*df*tfs(j)*cmplx(cos(2*pi*i*j*df*tau)
     *  ,sin(2*pi*i*j*df*tau))
      enddo
      s(i)=sum1
      enddo
      end
```

```
c--- Ce programme permet de calculer les sorities de RNA à chaque position de fenêtre le
long d'une trace.
c--- le signal lu est indiqué dans le programme soit tt(i,j)
c--  on peut ajouter du bruit par le programme bruitfilm.for ----------

c----- Paramètres d'entrées :
c lfenetre : la longueur de fenêtre utilisée.
c fichiers de lecture :  de la matrice tt

c ---- Paramètres de sortie :
c nom4 : qui contient les outputs de RNA à chaque position c de la fenêtre glissante (ipas)

          real tt(0:150,0:1000),attribu(0:1000,0:8),signal(0:1000)
          al wh(0:8,0:8),wo(0:8),oh(0:1000),oo(0:1000), oomax

          complex tfs(0:3750),s(0:3750)
          character*10 nom4
5     format(10a)
30    format(2x,'entrer la longeure de la fenetre lfenetre=..',$)
40    format(3x,'Entrer le fichier des outputs=..',$)

          write(6,30)
          read(5,*)lfenetre
          write(6,40)
          read(5,5)nom4

c-- Lecture de la section generée par le programme modele3.for----------
          open(10,file='tt')
          read(10,*)Ntrace
          read(10,*)m
      do i=1,Ntrace
      do j=0,m-1
          read(10,*)TT(i,j)
          enddo
      enddo
          close(10)

c ----Lecture des poids de RN calculé par Pythia----------
          open(10,file='wh.dat')
          do i=1,4
          do j=1,6
          read(10,*)wh(i,j)
      enddo
          enddo
          close(10)

          open(10,file='wo.dat')
          do i=1,4
          read(10,*)wo(i)
          enddo
```

```
        close(10)


c---- Calcul de l'output à chaque position de la fenêtre glissante---------
c------Lecture du signal ----------------

    tau=0.004
    do ipas=0,(m-lfenetre)
        write(6,*)'ipas=',ipas
        sum10=0.
        do ifenetre=0,lfenetre
c-------------- la trace itrace de la section TT ---------------
        signal(ifenetre)=TT(1,ifenetre+ipas)
        sum10=sum10+signal(ifenetre)
    enddo

        if (abs(sum10).gt.0.1) then
        call tf(signal,lfenetre,tau,nechf,tfs,sum6,sum8,sum9)
        call tfinv (tfs,lfenetre,tau,s)
        sum1=0.
        sum2=0.
        sum3=0.
        do i=0,lfenetre
        sum1=sum1+imag(s(i))
        sum2=sum2+sqrt(real(s(i))**2+imag(s(i))**2)
        sum3=sum3+atan(imag(s(i))/real(s(i)))
        enddo
        attribu(ipas,1)=sum1/lfenetre
        attribu(ipas,2)=sum2/lfenetre
        attribu(ipas,3)=sum3/lfenetre
        attribu(ipas,4)=sum6
        attribu(ipas,5)=sum8/lfenetre
        attribu(ipas,6)=sum9/lfenetre
    else
        attribu(ipas,1)=0.
        attribu(ipas,2)=0.
        attribu(ipas,3)=0.
        attribu(ipas,4)=0.
        attribu(ipas,5)=0.
        attribu(ipas,6)=0.
        endif
        enddo

        open(10,file='at.dat')
        do ipas=0,m-lfenetre
        write(10,*)attribu(ipas,1)
        write(10,*)attribu(ipas,2)
        write(10,*)attribu(ipas,3)
        write(10,*)attribu(ipas,4)
        write(10,*)attribu(ipas,5)
```

```
          write(10,*)attribu(ipas,6)
          enddo
      close(10)

c------ Calcul des Outputs par RNA ------------
      do ipas=0,(m-lfenetre)
          do ineurone=1,4
          sum11=0.
      do j=1,6
      sum11=sum11+attribu(ipas,j)*wh(ineurone,j)
          enddo
          oh(ineurone)=activ1(sum11)
          enddo

      sum12=0.
          do j=1,4
          sum12=sum12+oh(j)*wo(j)
          enddo
          oo(ipas)=activ1(sum12)
          enddo

c------ recherche du maximum de oo(ipas)---
          oomax=oo(0)
          do ipas=1,(m-lfenetre)
          if(oo(ipas).gt.oomax)then
          oomax=oo(ipas)
          endif
          enddo


          open(10,file=nom4)
          do i=1,lfenetre+4
          write(10,*)i,0
          enddo
      do ipas=0,(m-lfenetre)-(lfenetre/2-1)

      if(oo(ipas)/oomax.gt.0.9)then
      write(10,*)(ipas+13-(abs(16-lfenetre)/2)+(lfenetre/2))
          *,oo(ipas)/oomax

          else
          write(10,*)(ipas+13-(abs(16-lfenetre)/2)+(lfenetre/2)),0

          endif

          enddo

      close(10)

      end
```

```
      subroutine tf(s,nech,tau,nechf,tfs,sum6,sum8,sum9)
      dimension as(0:3750),ps(0:3750),s(0:3750),entropie(0:1000)
      complex tfs(0:3750)

c-- Caclcul de l'energie et de l'entropie
      sum6=0.
      do i=0,nech
      sum6=sum6+s(i)**2
      enddo
      sum7=0.

    do i=0,nech
      sum7=sum7+s(i)
    if(s(i).eq.0)then
      s(i)=0.01
    endif
      entropie(i)=sum7/sum6*log(s(i)**2/sum6)
      enddo
      sum8=0.

    do i=0,nech
      sum8=sum8+entropie(i)
      enddo
      sum9=0.
      do i=0,nech
      sum9=sum9+s(i)
      enddo
      close(10)
c-------------------------------------
      pi=atan(1.)*4
      fn=1/(tau)
      df=fn/nech
      nechf=nint(fn/df)

      do j=0,nechf
      sum1=0.
      sum2=0.
      do i=0,nech
      sum1=sum1+s(i)*cos(2*pi*j*i*tau*df)
      sum2=sum2+s(i)*sin(2*pi*j*i*tau*df)
      enddo
      as(j)=sqrt(sum1**2+sum2**2)
      ps(j)=atan(-sum2/sum1)

      tfs(j)=2*tau*cmplx(as(j)*cos(ps(j)),as(j)*sin(ps(j)))*
   *  cmplx(as(j)*cos(ps(j)-pi/2.),as(j)*sin(ps(j)-pi/2.))
      enddo
      end
```

```
      subroutine tfinv (tfs,nech,tau,s)
      complex tfs(0:3000),s(0:3000)
      complex sum1
 pi=atan(1.)*4
      fn=1./(tau)
      df=fn/nech
      nechf=nint(fn/df)
 do i=0,nech
      sum1=cmplx(0,0)
      do j=0,nechf
      sum1=sum1+(1./nech)*df*tfs(j)*cmplx(cos(2*pi*i*j*df*tau)
*,sin(2*pi*i*j*df*tau))
      enddo
      s(i)=sum1
      enddo
      end
      real function activ1(x)
      activ1=1/(1+exp(-4*(x-0.5)))
      end function
```

```
c --- Programme de la technique des moments invariants ---------
c     pour la détection des réflecteurs sismiques.

c----------------------- Les paramètres d'entrée -----------------------------
c     'rik1.dat' riker à 17 echantillons
c     lfenetre:la longeur de la fenetre.
c     nk : l'ordre du moment utilisé.
c     itraces : la trace selectionnée du film nom1.
c----------------------- Les paramètres de sortie ---------------------------
c     mrik : la matrice de référence.
c     eqm.dat: erreur quadratique moy.  le long de la trace traitée.
c     eqmn.dat : erreur quadratique moy normalisée le long la trace traitée.

      character*10 nom1
      real tt(0:100,0:600),momts(0:100,0:600,0:10),moment(0:10)
      real s(0:500)
      real reference(0:10),EQM(0:100,0:600)
5     format(10a)
10    format(1x,'entre le film nom1=..',$)
20    format(1x,'entrer le N° de la trace itraces=...',$)
30    format(1x,'entrer la longeur de la fenetre lfenetre=...',$)
40    format(1x,'entrer l''ordre du moment nk=...',$)
            write(6,10)
            read(5,5)nom1
            write(6,20)
            read(5,*)itraces
            write(6,30)
            read(5,*)lfenetre
            write(6,40)
            read(5,*)nk

      open(10,file=nom1)
            read(10,*)Ntrace
            read(10,*)m
      do i=1,Ntrace
      do j=1,m
            read(10,*)TT(i,j)
            enddo
      enddo
            close(10)

            do itrace=1,ntrace
           do ipas=1,(m-lfenetre)
            do ifenetre=1,lfenetre
c--------------- la trace itrace de la section TT ---------------
            s(ifenetre)=TT(itrace,ifenetre+ipas)
            enddo
            call momoents(lfenetre,s,nk,moment)
            do k=0,nk
            momts(itrace,ipas,k)=moment(k)
```

```
          enddo
          enddo
          enddo
c---- Caclul de la matrice référence ------------------------
          open(10,file='rik1.dat')
          nechr=17
      do i=1,nechr
          read(10,*)a,s(i)
          enddo
          close(10)
          call momoents(nechr,s,nk,reference)
          open(10,file='mrik.dat')
          do k=1,nk
          write(10,*)k,reference(k)
          enddo
          close(10)

c----- Calcul d EQM à chaque position de la fenêtre -------------
          do itrace=1,ntrace
      do ipas=1,(m-lfenetre)
          sum4=0.
      do k=1,nk
      sum4=sum4+(momts(itrace,ipas,k)*reference(k))
      enddo
          EQM(itrace,ipas)=sum4
          enddo
          enddo

c------ EQM pour une seule trace qui est itraces---------
          open(10,file='eqm.dat')
          do ipas=1,lfenetre
          write(10,*)ipas,EQM(itraces,1)
          enddo
          do ipas=1,(m-lfenetre)
          write(10,*)ipas+lfenetre,EQM(itraces,ipas)
          enddo
          close(10)
c----- Normalisation du EQM ----------
          eqmmax=0.
          do ipas=1,(m-lfenetre)
          if(EQM(itraces,ipas).gt.eqmmax)then
          eqmmax=EQM(itraces,ipas)
          endif
          enddo

          open(10,file='eqmn.dat')
          do ipas=1,lfenetre
          write(10,*)ipas,EQM(itraces,1)/eqmmax
          enddo
          do ipas=1,(m-lfenetre)
```

```
      write(10,*)ipas+lfenetre,EQM(itraces,ipas)/eqmmax
      enddo
      close(10)
      write(6,*)'  '
      write(6,*)'  '
      write(6,*)' Les résultats se trouvent des les fichier '
      write(6,*)'  '
      write(6,*)'eqm.dat : inter correlation. le long de la trace'
      write(6,*)'  '
      write(6,*)'eqmn.dat :inter correlation moy. normalisée le long
*     de la trace'
      write(6,*)'  '
      write(6,*)' appuyer sur un touche pour terminer'
       pause 2
      end
 subroutine momoents(nech,s,nk,moment)
 real moment(0:100),s(0:1000)
 sum1=0.
 do i=1,nech
      sum1=sum1+s(i)
      enddo
 moment(0)=sum1/nech
 do k=1,nk
      sum2=0.
      do i=1,nech
      sum2=sum2+(i**k)*((s(i)-moment(0))**k)
      enddo
      moment(k)=sum2/nech
      enddo
      end
```

Bibliographie

BIBLIOGRAPHIE

[1] Benhama A., 2000, Traitement sismique, IAP/ Sonatrach, Boumerdès.

[2] Carlos Calderon-Macias, Mrinal K. Sen and Paul L. Stoffa, 2000, *"Artificiel Neural Networks For Parameter Estimation in Geophysics "*, *Geophysical Prospecting*, 48,21-47.

[3] Carlos.E.Mejia, 1992, Architecture Neuronale Pour l'Approximation des Fonctions de Transfert : Application à la télédétection, Thèse de doctorat, Université de Paris Sud Centre d'Orsay.

[4] Chunduru R. K., Sen M. K., Soffa P.L, *" 2D Resistivity Inversion Using Spline Parametrisation and Simulated anneanling"*, 1996, Geophysics, 61, 151-161.

[5] Guerchaoui A., 1999, *« Time Delay Estimation »*, A Tutorial, Séminaire sur les Techniques et Architectures des Système de Détection.

[6] Guerchaoui A. and Benhama A., 2000, *"Automatic Picking Of Seismic reflections. Two Methods Using an Energy Criteria"*. 62 EAGE conference, Glasgow.

[7] Koefoed O., 1979, Geosounding Principles, 1: Resistivity Sounding Measurements, Elsevir Science Publishing Co.

[8] IDIR S., 2000, Etude comparative des Technique de Pointé et des Suivis des horizons sismiques, mémoire Ingénieur, IAP/Sonatrach, Boumerdès

[9] Ulrich P., 1999, *"Horizon Tracking in 3D Seismic Datasets Using Neural Networks"*, EAGE 61st conference and technical exhibition, Helsinki Finland.

[10] Telford W.M, Geldart L.P., Sheriff R.E., 1998, *"Applied Geophysics"*, Cambridge, U.K.

[11] Mari J-L., Glangead F., Coppens F., 1997, Traitement du signal pour géologues et géophysiciens, Technip, Paris, France.

[12] Coulon F. De., 1996, Théorie et traitement des signaux, Dunod

[13] Lasfargues P., 1957, Prospection électrique par courants continus, Masson & C^{ie,} , Paris, France.

[14] Corsini M.M, 1997, Introduction aux réseaux de neurone, Cours université Victor Segalen , Bordeaux, France.

[15] Jimmy S., Manoj C., Singh S.B., 2004, "*A direct inversion scheme for deep resistivity sounding data using artificial neural networks",*Earth Planet Science,113,N° 1,pp.49-66.

[16] MEUNIER J., 2003, Cours Reconnaissance de Forme, Université de Monreale Canada.

[17] Dittmer J., Szymanski J., 1995, *The stochastic inversion of magnetic data using simulated annealing algorithm*. Geophysical Prospecting, 43, PP. 397-416.